NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

FACILITATING SECURE MAIL IN A HIGH ASSURANCE LAN

by

Emma J. M. Brown

September 2000

Thesis Advisor:	Cynthia E. Irvine
Second Reader:	James P. Anderson

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 2000	Master's Thesis

4. TITLE AND SUBTITLE:
Facilitating Secure Mail In a High Assurance LAN

5. FUNDING NUMBERS

6. AUTHOR(S)
Brown, Emma J. M.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (maximum 200 words)

Currently, almost all DoD systems are operated at a single level, classified or unclassified. The problems encountered on these single level systems with mail exchange, its storage, and manipulation are the multiple networks and workstations required to handle different security levels of data as well as the high cost of maintaining them. The Naval Postgraduate School Multilevel Secure Local Area Network (MLS LAN) project supports a high assurance server. This LAN is COTS-driven (commercial-off-the-shelf) and enforces a mandatory security policy while permitting users to employ standard office productivity tools on standard workstations. Initially, there was no means for multilevel mail exchange between clients of the system.

This research was to implement the simple mail transfer protocol (SMTP) server, Sendmail, on the Wang Federal XTS 300 as a multilevel server. A port of a UNIX version of Sendmail 8.9.3 was made to the XTS 300. Modifications to Sendmail were required so that it could be supported by the UNIX-like XTS 300 STOP 4.4.2 operating system. Sendmail proved to be a successful mail server for exchange of mail between system clients. Tests demonstrated successful transmission of simple mail and mail with attachments.

14. SUBJECT TERMS
Multilevel Security (MLS), MLS Local Area Network (LAN), High Assurance, Sendmail, Commercial-off-the-shelf (COTS), platform, server, client, trusted path, Trusted Computing Base (TCB), Wang Federal XTS 300

15. NUMBER OF PAGES 100

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

FACILITATING SECURE MAIL IN A HIGH ASSURANCE LAN

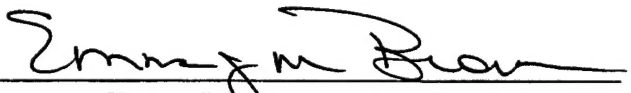
Emma J. M. Brown
Lieutenant, United States Navy
B.S., Savannah State College, 1993

Submitted in partial fulfillment of the
requirements for the degree of

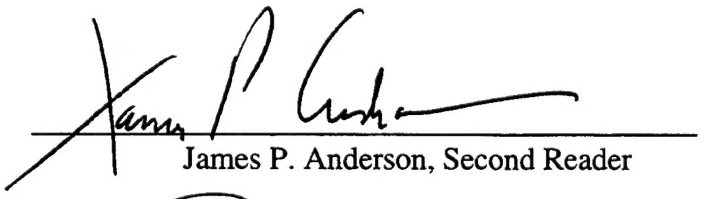
MASTER OF SCIENCE IN SYSTEMS ENGINEERING

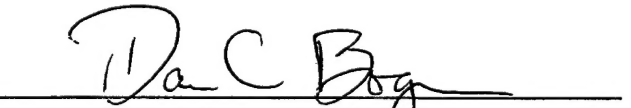
from the

NAVAL POSTGRADUATE SCHOOL
September 2000

Author: 
Emma J. M. Brown

Approved by: 
Cynthia Irvine, Thesis Advisor


James P. Anderson, Second Reader


Dan C. Boger, Chairman
Information Warfare Academic Group

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Currently, almost all DoD systems are operated at a single level, classified or unclassified. The problems encountered on these single level systems with mail exchange, its storage, and manipulation are the multiple networks and workstations required to handle different security levels of data as well as the high cost of maintaining them. The Naval Postgraduate School Multilevel Secure Local Area Network (MLS LAN) project supports a high assurance server. This LAN is COTS-driven (commercial-off-the-shelf) and enforces a mandatory security policy while permitting users to employ standard office productivity tools on standard workstations. Initially, there was no means for multilevel mail exchange between clients of the system.

This research was to implement the simple mail transfer protocol (SMTP) server, Sendmail, on the Wang Federal XTS 300 as a multilevel server. A port of a UNIX version of Sendmail 8.9.3 was made to the XTS 300. Modifications to Sendmail were required so that it could be supported by the UNIX-like XTS 300 STOP 4.4.2 operating system. Sendmail proved to be a successful mail server for exchange of mail between system clients. Tests demonstrated successful transmission of simple mail and mail with attachments.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND

Department of Defense systems continue to experience a problem with not only providing control of access to and movement of data based on sensitivity levels, but also with preserving the compatibility with commercial-off-the-shelf (COTS) application software. Often, COTS applications have priority, so independent systems at each access class are constructed and set to system high conditions. This causes sharing techniques to become expensive and inefficient in terms of equipment, space, and administration. To address this problem, the Naval Postgraduate School Center for Information Systems Security Studies and Research (NPS CISR) faculty, staff, and students built the multilevel secure (MLS), COTS-driven Local Area Network (LAN).

The MLS LAN provides multilevel secure services to its clients while allowing them to use standard office productivity tools and software on standard personal computer (PC) workstations. The MLS LAN uses the Wang XTS 300, a high assurance server that is multilevel secure but not very user-friendly. Since the XTS 300 is costly and cannot be used at every desktop, it is used as a server connected to inexpensive workstations via a LAN. The COTS

workstations and software give the user a familiar, easy to use interface to the services of the XTS 300. The XTS 300 allows the user to login at different access levels from the client workstations. The XTS 300 provides security policy enforcement over all data provided to clients. No data are stored on client machines.

The research area of this thesis was to determine a method whereby multilevel mail could be exchanged between clients of the MLS LAN. The primary focus was to implement Sendmail on the XTS 300 as a LAN-based server and to design configuration and implementation procedures, which facilitate the exchange of mail between its clients across multiple security levels.

B. OBJECTIVE

The objective of this research is to facilitate multilevel mail exchange between clients on the MLS LAN that employ standard office productivity applications on standard workstations. This mail service will be achieved by running the mail facilities on the XTS 300 server. The XTS-300 is a Pentium-class platform running an UNIX-like operating system known as STOP 4.4.2. Though it is not very straightforward to use, it supports high assurance security policy enforcement. The approach in this thesis was to configure

Sendmail for the XTS 300 to provide mail services on the server. Next, this service needed to be integrated with the standard mail services found on the LAN workstations so that mail could be sent and received via the workstation while adhering to the overall functional and security requirements of the MLS environment. Since the user employs standard workstations and productivity software to send mail within the MLS LAN, ease of use for mail exchange increased dramatically.

C. SCOPE

The extent of this research incorporates modifications to and configuration of Sendmail required to achieve mail exchange between clients of the MLS LAN, the degree of movement of mail achievable through that configuration, to both internal and external addresses was examined. Some of the research questions that will be answered are:

- What mail facilities are already available in the system?
- What communications are required between the client and the server at the application level?
- What support does IMAP provide?
- How is mail to be sent between clients managed and how is it encrypted and protected?
- What needs to be done to transmit mail outside of system (Internet mail)? What extensions or

additional facilities are required in order for Sendmail to support Internet mail?

- What are the design and implementation considerations for system components, so that the MLS environment can be used to greatest advantage?

- Will both unlabeled and labeled mail be permitted?

- What are the protocols for receipt, retransmission, and acknowledgement of mail in the MLS environment?

- What are the implications of sending mail from low to high users or from unlabeled users to labeled users?

- What is required in terms of communications server support for MLS mail services to external destinations beyond the MLS LAN?

II. EXCHANGING MAIL BETWEEN CLIENTS IN AN MLS ENVIRONMENT

A. DEFINITION OF PROBLEM

Currently, almost all DoD classified systems are operated at a single level. The level of the system is set to the highest level of any data that is being processed within the system. This is known as system-high mode, and it requires everyone who uses it to hold clearances at the highest security level of any data on the system. All outgoing mail from that system must be handled at the highest level which creates many inefficiencies: the entire system, including the server, must be maintained in a secure environment; additional systems must be provided to allow users to send mail at lower classifications; and separate networks must be managed for each access class.

These separate networks are required in the typical systems due to Trojan Horses that may exist in application software. A Trojan Horse is the term given to hidden malicious functionality embedded in application software. A Trojan Horse could allow data of a higher sensitivity level to be passed to a lower sensitivity level without the user's knowledge.

The MLS LAN resolves problems of multiple networks and workstations to handle different security levels of data as

well as the high cost of maintaining them through the use of the high assurance multilevel server. A highly trusted server, the Wang XTS 300, enforces security policy for all creation of, movement of, and access to mail. The XTS 300 permits controlled sharing of sensitive data by users at multiple security levels. Through the use of COTS application-level software, the data stored on the XTS 300 can be exchanged between clients as allowed by policy.

There are three basic services we usually associate with mail [1]: user agent (UA), message transport agent (MTA), and a message store (MS). The user agent is the application used to create and read mail messages, for example, Microsoft Outlook, Eudora, or on UNIX systems, Mail and Rmail. A message transfer agent is the service that actually sends the mail message from one system to another, such as Sendmail. The message store is a system for storing messages when the user agent is not connected to the message transfer agent. The most common message store is currently Post Office Protocol, Version 3 (POP3). Currently, there is no feature installed on the MLS LAN, which permits exchange of mail between LAN-based clients of the XTS 300 server. The only facilities available on the XTS 300 server are the UNIX-based user agents, Mail and Rmail. These facilities do not provide services for LAN-based clients and are only

accessible to users logged in on serial ports attached to the XTS 300. Basically, these facilities treat the XTS 300 as if it were an old-fashioned mainframe. By using Sendmail, a Mail Transport Agent (MTA), mail exchange between clients of the MLS LAN could be achieved via user-friendly, commercial software applications such as Microsoft Outlook or a mail tool associated with a network browser such as Netscape. Sendmail is a UNIX-based, highly specialized program that delivers mail and transports it between machines, like a post office.

B. MLS LAN

The MLS LAN provides organizations with a cost-effective, multilevel, easy-to-use office environment leveraging existing high assurance technology. The MLS LAN provides a networking environment that provides concurrent high assurance access for network users to multiple sensitivity level data through the incorporation of inexpensive commercial personal computers. To ensure positive control over the communications between MLS LAN entities, certain connection protocols are required. An illustration of how these protocols facilitate the MLS LAN is depicted in Figure 1. [2]

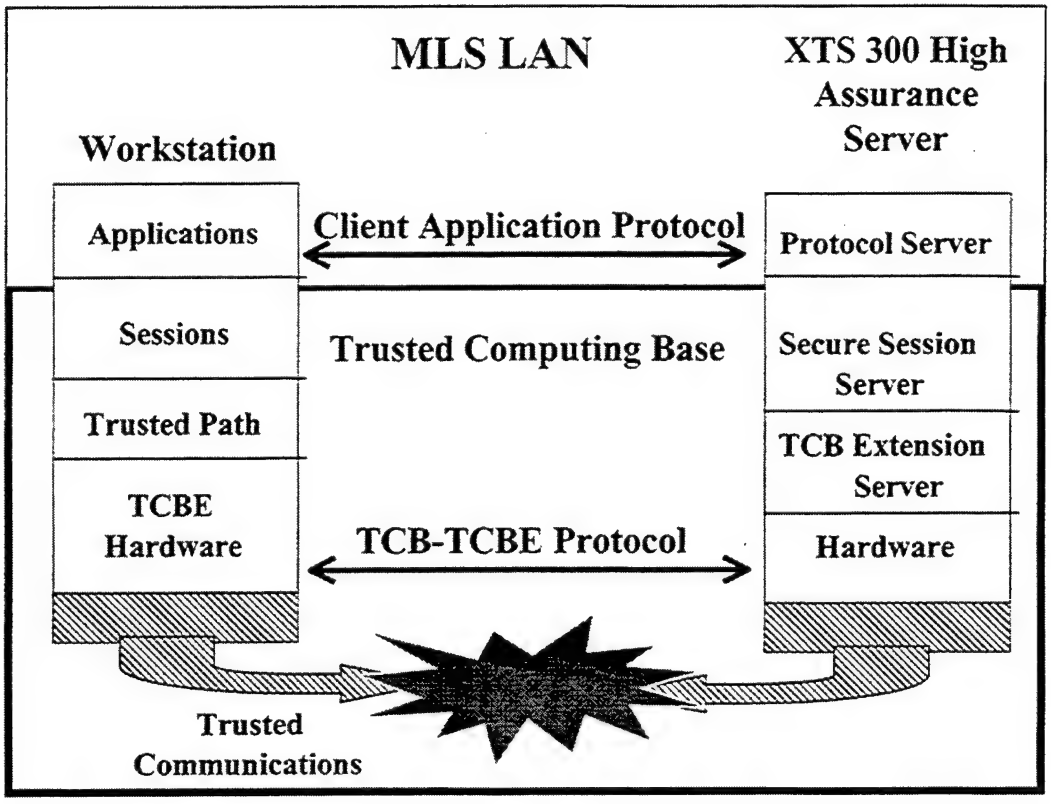


Figure 1. MLS LAN

The Trusted Computing Base (TCB) must provide protection against disclosure and modification of information on all transmissions between components of the MLS LAN in order to provide the high assurance required throughout the network. This is accomplished through the establishment of a non-avoidable Trusted Communications Channel that provides mutual authentication for the two TCB entities and data encryption on all transmissions between them. This Trusted Communications Channel thus presents the

protected conduit through which all other MLS LAN protocols may negotiate connectivity [2].

Access to the MLS LAN is controlled by the establishment of a session, which requires user authentication to the Trusted Computing Base. This operation or any other security-related operations between the client and the TCB must be conducted through a trusted path. This requirement is established in the Trusted Computer Security Evaluation Criteria (TCSEC), section 36.3.2.1.1 (Trusted Path) which states:

> "The TCB shall support a trusted communications path between itself and users for use when a positive TCB-to-user connection is required (e.g., login, change subject security level). Communications via this trusted path shall be activated exclusively by a user of the TCB and shall be logically isolated and unmistakably distinguishable from other paths" [4].

It is also required by the Common Criteria for Information Technology Security Evaluation Version 2.1, under the Trusted Path class (FTP). The Common Criteria states:

> "Absence of a trusted path may allow breaches of accountability or access control in environments where untrusted applications are used. These applications can intercept user-private information such as passwords, and use it to impersonate other users. As a consequence, responsibility for any system actions cannot be reliably assigned to an accountable entity. Also, these applications could output erroneous information on an unsuspecting user's display, resulting in subsequent user actions that may be erroneous and may lead to a security breach." [5]

The Common Criteria specifically designates a Trusted Path family (FTP_TRP) for communications between the user and the TCB for use during all security related operations dealing with the establishment, modification, and termination of a session.

> "This family defines the requirements to establish and maintain trusted communications to and from users and the TSF (Target of Evaluation Security Functions). A trusted path may be required for any security-relevant interaction. Trusted path exchanges may be initiated by a user during any interaction with the TS, or the TSF may establish communications with the user via a trusted path." [5]

These trusted path communications are supported by the TCB-to-TCBE connection protocol. Following session establishment the MLS LAN client will be authorized to conduct normal operations within the MLS LAN environment including network application services. The security of these network services connections are ensured by protection

of application service requests transmitted from the client to the Secure Session Server. The Secure Session Server (SSS) will validate the user's session sensitivity level and access level. If the SSS authorizes the user, it will create a socket interface to the Application Protocol Server (APS) and allow application operations to commence. [6]

C. MLS LAN CURRENT MAIL CONFIGURATION

The MLS high assurance multilevel mail server uses the Trusted Computing Base (TCB) supported by the XTS 300 to enforce system security policy on untrusted instances of the mail server program at each classification level. A TCB is the combination of protection mechanisms within a computer system, including hardware, firmware, and software, which is responsible for enforcing a security policy [2]. The TCB creates a basic protection environment and provides additional user services required for a trusted computer system. The Internet Message Access Protocol, Version 4 (IMAPv4) is implemented on the XTS 300. The TCB allows the placement of a large untrusted mail server program on the system without having to establish its correctness with reference to security policy and also allows the current evaluation of the system against the Trusted Computer Systems Evaluation Criteria to remain in effect [7].

IMAP [8] is a mail message store (MS) service, and it maintains all mail at the server. No mail messages are actually stored on the workstation, only on the server. Incoming mail is moved by IMAP from a spool to a user's "Inbox." Users can read, delete, and manage their mail. Messages to be saved can be stored in mail folders at the server. Server-based storage is an advantage of IMAP over other message store protocols such as POP3 since no classified material is stored at the workstation. [7] In order to provide access to data at multiple security levels, the messages are stored on the XTS 300, which provides the necessary assurance of enforcement of the system's mandatory and discretionary security policies. An added advantage of storing the messages on the server and not on the workstation is that the user can login from any workstation connected to the server and manage mail.

D. OVERVIEW OF SENDMAIL

1. Typical Mail Services

As discussed previously, a mail user agent (MUA) is a program that a user runs to read, reply to and dispose of mail. In a distributed system like the MLS LAN, the user uses a MUA such as Microsoft Outlook or Netscape Communicator to create a mail message on the client

workstation. When the client "sends" the mail message, the MUA invisibly passes the mail message to the Sendmail program, a mail transport agent (MTA), for delivery to the recipient. Sendmail then puts the mail message onto a spool or queue. When the recipient requests new mail using the MUA on the client workstation, IMAP, the Message Store (MS) takes the mail message that Sendmail has on its spool and puts it into the client's "Inbox". The client can then read its "Inbox" and either delete or store the message in a folder, which resides on the server but is created via commands issued from the client machine. This is achieved through the cooperation of the MUA and MS programs that are running on the client and server respectively. Figure 1 illustrates how mail is managed by mail services.

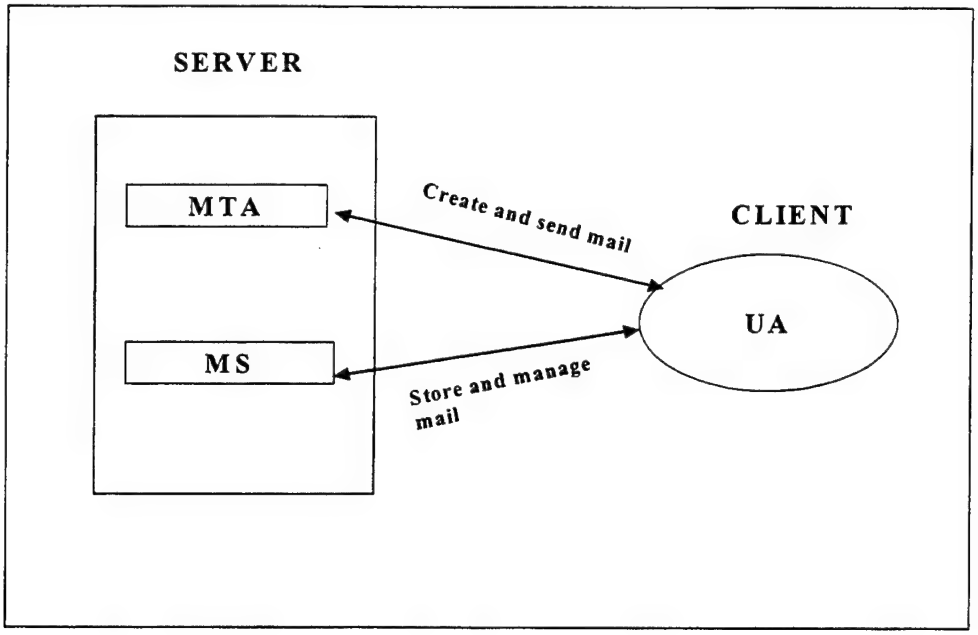


Figure 2. Typical Mail Services

2. Sendmail 8.9.3

The version of Sendmail that was ported to the Wang XTS 300 is Sendmail 8.9.3. The Sendmail program is actually composed of several parts, including programs, file directories, and the services it provides. A configuration file defines the location and behavior of these other parts and contains rules for rewriting addresses. A "queue directory" holds mail until it can be delivered. An "alias file" allows alternative names for users and creation of mailing lists [1].

The configuration file is the foundation of Sendmail; without it Sendmail cannot function. It contains information including file locations, permissions, and modes

of operation. Also, the configuration file contains rule sets, which convert a mail address into another form that may be required for delivery. These rules are designed to allow fast processing by Sendmail and therefore appear unintelligible to someone other than the programmer. For example:

```
R$+@$+      $:$1<@$2>  focus on domain
```

But what appears complex is really just concise. Here, R labels this line as a rewriting rule, and the "$+" expressions mean to match one or more parts of a mail address [1]. These rewriting rules within the configuration files enable aliasing and forwarding of mail messages. With experience, such expressions soon become meaningful.

Since not all mail messages can be delivered immediately, Sendmail must be able to store them for later transmission. Sendmail has a queue directory just for this purpose. A mail message will be held in this queue directory under the following conditions [1]:

- When the destination is down or not reachable
- When a mail message has many recipients

- When a mail message is expensive. If a mail message is sent over a long-distance line, then the mail message may be queued for delivery when the rates are lowest.

- When safety is of concern due to machine crashes, etc.

Aliases form the foundation for mailing lists. They allow mail sent to one address to be delivered to another address. The aliases file is the heart of aliasing and is often stored in a database format. The aliases file allows alternative names for users and creation of mailing lists.

E. XTS 300

It is important to discuss how the Wang Federal, Incorporated XTS 300 system is organized before continuing. The XTS 300 includes the STOP 4.4.2 [9] operating system and commercially available hardware products. STOP 4.4.2 is a product of Wang Government Services, Incorporated. The hardware includes the Intel Pentium processor, hard disk, floppy disk drive, SCSI adapter, Ethernet card, streamer tape drive, and keyboard. [9]

The XTS 300 system supports both a mandatory sensitivity policy and a mandatory integrity policy. It provides 16 hierarchical sensitivity levels, 64 non-hierarchical sensitivity categories, 8 hierarchical

integrity levels, and 16 non-hierarchical integrity categories. Some of the hierarchical integrity levels are used by the system for role separation and the others are available to enforce user-related policies. The combination of mandatory sensitivity and integrity hierarchical and non-hierarchical levels is called the Mandatory Access Control (MAC) label. The system also supports a discretionary access control policy. [9]

The primary software components of the XTS 300 are the Security Kernel, TCB System Services (TSS), Trusted Software, and the Commodity Application System Software (CASS) [9]. The Security Kernel provides basic operating services and enforces system security. The TCB System Services software provides general trusted services to XTS 300 application and system software. Trusted software provides additional security services outside the Security Kernel.

The Security Kernel enforces the mandatory security policy and also manages resources, scheduling, interrupts, and auditing. The security policy consists of rule sets governing the system security and system integrity. The security rules protect data from unauthorized access, while the integrity rule sets protect the data from unauthorized access. [9]

CASS provides an environment on the XTS 300 for the execution of UNIX-based application programs. Although CASS is untrusted, it is considered high-integrity system software. It is protected from modification by application software by the ring mechanism enforced by the Security Kernel. That is, CASS provides untrusted operating system services to application software. Though it is not part of the TCB on the XTS 300, it is constrained by the security policy of the TCB [4].

The policy that the XTS 300 TCB enforces is the DoD policy on multi-level secure computing as formalized in the National Computer Security Center (NCSC) approved Bell-LaPadula mathematical model [10]. Specifically, the TCB enforces the following mandatory security rules:

- Simple security — a user is allowed to read (or execute) a data object (e.g., file) only if the security level of the user dominates that of the object.
- Security *-property - a user is allowed to write a data object only if the security level of the object dominates that of the user.

The XTS 300 TCB also includes an integrity policy formalized by the Biba Model [11], which enforces the following mandatory integrity rules:

- Simple integrity - a user is allowed to read or execute a data object only if the integrity level of the object dominates that of the user.

- Integrity *-property - a user is allowed to write a data object only if the integrity level of the user dominates that of the object.

The XTS 300 enforces a discretionary or need-to-know policy wherein access to an object is determined by the identity of the users and/or groups to which they belong. Specifically, a user is only allowed access to a data object in the mode(s) granted by the owner of the object. Each object has allowed permissions (read, write, execute) for the members of the owner's group, for other specifically identified users and groups, and for all others.

The algorithm used by the TCB to determine whether a user should be granted discretionary access to an object is as follows [9]:

- If the user is the owner of the object, use the specified owner permissions;

- If there is an entry for the user in the Access Control List (ACL), use the permissions contained in the ACL entry;

- If the user's current group is the same as the group ownership of the object, use the specified group permissions;

- If there is an entry for the user's current group in the ACL, use the permissions contained in the ACL entry;

- Use the specified other (world) permissions.

The XTS 300 also enforces a general and configurable policy that strengthens the traditional mandatory and discretionary access rules [9]. This enforcement policy limits access to objects based on their subtype. Specifically, a user is allowed to access a data object only if the subtype of the object is present on the user's process' accessible subtype list for that object type. The accessible subtype list may not be modified by the user. Also, the subtype of an object specified by the creator of that object and must be taken from the creating user's accessible subtype list for that object type. The subtype of an existing object may not be modified.

A ring mechanism is also provided to augment the security of the XTS 300 system [9]. It is used to isolate portions of a process from tampering. Ring 0 is reserved for the Kernel and is the most privileged; ring 1 is reserved for the TCB System Services; ring 2 is reserved for Trusted Software, CASS, and site-developed trusted processes, and is less privileged; and ring 3 is reserved for user processes and is the least privileged. A process may access information residing in a ring of the same or lesser privilege, but not in a ring of greater privilege.

To support all of the XTS 300 security requirements, the TCB (together with the hardware) mediates all requests to access data. Data can be thought of as passive objects being accessed by active subjects. There are five types of objects supported by the TCB [9]:

- Processes
- Devices
- File system objects (files, directories, etc.)
- Segments (temporary shared, temporary private, and shared memory
- Semaphore sets

Processes may also be subjects, as they may access object data.

Each object is referenced by its own identifier, and each has its own set of access and status information. While status information varies depending on the object type, access information is common to all objects. The access information associated with an object includes its mandatory and discretionary access attributes, and is the basis upon which the TCB makes security decisions. An object's mandatory access information consists of: security level and categories, integrity level and categories. The discretionary access associated with an object includes [9]:

- Object's owner and group identifiers

- Read, write, and execute permission for owner, members of the group, and other users
- An access control list (ACL) consisting of up to seven user and group identifiers and their specific permissions (read, write, execute)
- The object's subtype (subtypes cannot be used for shared memory segments and semaphore sets)

The following figure is a depiction of the Wang Federal, Inc. XTS 300 system [9].

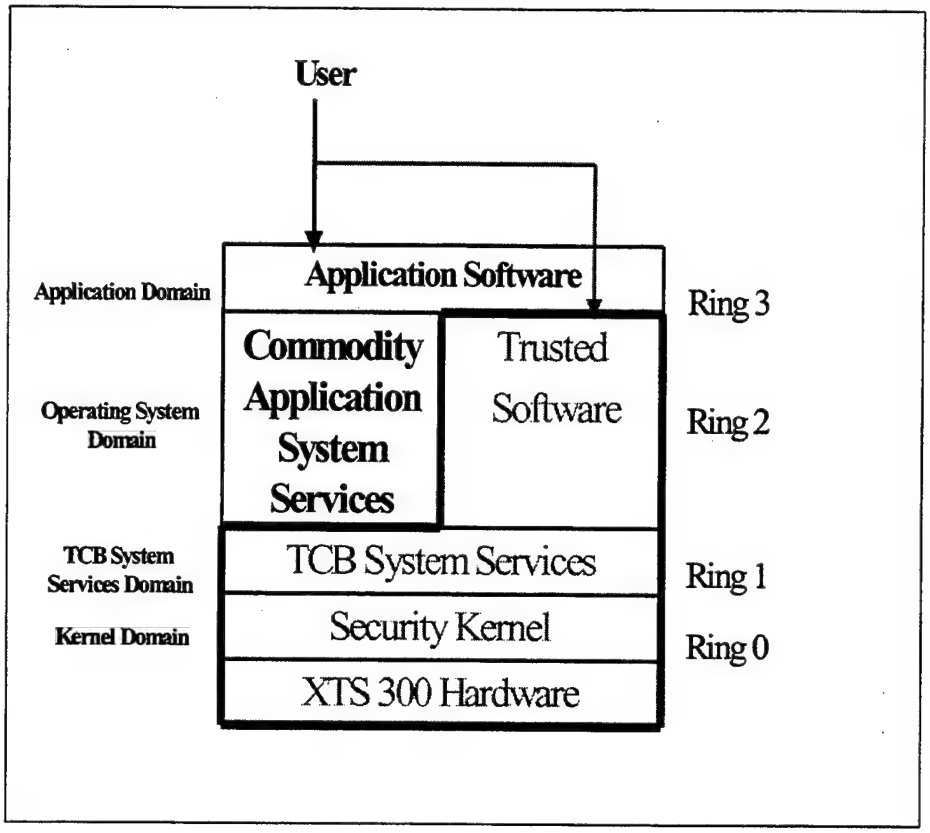


Figure 3. XTS 300 System [9]

THIS PAGE INTENTIONALLY LEFT BLANK

III. APPLICATION OF SENDMAIL 8.9.3 TO THE XTS 300

A. PORTING SENDMAIL TO THE XTS 300

The Sendmail open-source, UNIX-based software was obtained online from Sendmail.org. After the software was unzipped and untarred, it was downloaded to the XTS 300. Eric Allman originally wrote Sendmail while he was still a student at the University of California at Berkeley. At the start of this research, Sendmail 8.9.3 was the latest available version of the program. The software was designed to be used on single-level systems and includes all of the source code needed to compile it.

Because of the uniqueness of the STOP 4.4.2 operating system of the XTS 300, the port required several changes to the source code. The Sendmail source code package came with compiling/building instructions for several versions of UNIX. Since the XTS 300 and its operating system, STOP 4.4.2, though close to UNIX, are quite unique, those instructions were inapplicable for this port. After several modifications to the Build and Makefile source codes files and several trials and errors, the Sendmail program compiled with zero errors.

B. SPECIFICATION

The Sendmail program was placed on the XTS 300 system without having to establish its correctness because the TCB supported by the XTS 300 constrains untrusted instances of the mail server program at each classification level. This allowed the current evaluation of the system against the Trusted Computer Systems Evaluation Criteria to remain in effect.

The mail protocol that was used with Sendmail is Simple Mail Transport Protocol (SMTP). A protocol is the format of the data stream between two cooperating processes in a network [1]. SMTP is built into Sendmail, and the official reference for SMTP is Request for Comments (RFC) 821, as modified by RFC 1123 and others.

The following are requirements for the Sendmail server:

- It must support a COTS PC client and unmodified COTS mail client software
- It should implement the standard SMTP functions as established by RFC 821 and Internet mail headers described in RFC 822.
- Modifications that make the mail server work at multiple security levels should not affect the execution of client mail software such as Microsoft Outlook, Netscape Communicator, and Pine.
- A client operating at a particular sensitivity level should be able to send and receive all mail dominated by that sensitivity level.

- A client operating at a particular sensitivity level should only be able to append to and send mail at that particular sensitivity level

C. MODIFICATION REQUIREMENTS

STOP 4.4.2 required that some basic configuration changes be made to the Sendmail source C-code so that it could be compiled on the XTS 300. The changes that were made are contained in Appendix A.

The first changes were incorporated so that the STOP 4.4.2 operating system could process certain UNIX commands. The commands of the UNIX-based source code were changed to specific commands of the XTS 300's operating system. Also, before the source code could be compiled, the code had to be modified for the unique file architecture of the XTS 300. Therefore, special link commands added to the "Build" file were necessary so that the program could find the files it required to compile successfully. These added commands are found in Appendix A.

Next, it became necessary to define particular header files, standard system and function calls, and library files which the XTS 300 operating system did not recognize, but which were required by the source code. In most cases, just simply creating empty header files, removing undefined

functions, and inserting XTS-specific functions and library files resolved these problems.

Once the task of debugging the source code was complete and the code successfully compiled with zero errors, the Sendmail source files had to be prepared for execution on the XTS 300. Specifically, the configuration file which is contained in Appendix B was written for the MLS LAN configuration which is currently only for local delivery. A simple configuration file was obtained from Frederick Avolio and Paul Vixie's book, "Sendmail, Theory and Practice" [12], and was modified to incorporate site specific parameters. Since this system is used only for local delivery, the sendmail.cf was made rather easily following the README.cf files that came with the Sendmail source code.

The Sendmail configuration file, generally named sendmail.cf (Appendix B), contains several classes of information that determines the behavior of Sendmail on a host system.

- Options determine the values of numerous Sendmail parameters (for instance, file and directory paths, operational control switches, timeout values)
- Header definitions are templates used to specify required and optional message headers and their formats
- Mailer definitions specify the programs that will be used to deliver various kinds of mail(for instance,

local delivery, delivery to a file or program) as well as specifying details of Sendmail's interaction with them

- Macro and class definitions provide names for strings and sets of strings (for instance, domain name of host, set of alternate names)

- Rewriting rule sets are used to parse and transform addresses. In addition to controlling the appearance of addresses and directing special handling of certain classes of addresses, rewriting rules are used by Sendmail to determine, for each message recipient, the final delivery address, the mailer to use and the host system where the message should be delivered (or relayed).

- Key (map) file declarations specify the path and other attributes of files that can be used in rewriting rules to lookup and transform elements in an address.

Next, input/output commands of the source files had to be changed so that they were XTS 300-specific. Several commands, including getchar, fgets, putchar, fflush, stdin , stdout, select, etc., all of which are standard UNIX commands, were modified so that communications connections to the XTS 300's Application Protocol Server could be established. This process was very time-consuming and required much trial, error and investigation before all of the input/output commands were found and modified as necessary.

Finally, Sendmail was ready to run from the command line of the XTS 300. Again, trying to execute Sendmail was

difficult in the beginning. Debugging was turned on through manual modifications of some of the source files as well as commanding Sendmail itself from the command line.

The first tests were run using Mail, which is a user agent, (UA). Preparing and reading mail messages is done with a UA and includes programs such as Eudora, Pine, Mailx, Mail, etc., as discussed previously. Delivering mail messages is generally handled by Mail Delivery Agents (MDA) or Message Stores (MS). These programs generally do one type of delivery, such as putting mail into a local mailbox file. An example is IMAP. Sendmail does forwarding of the mail message via SMTP. Sendmail is a Mail Transport Agent (MTA) and determines how a message has to be routed to get to a recipient. It accepts mail from other MTAs and relays it to an agent closer to the ultimate recipient; it handles the interpretation of address aliases; it transforms addresses so that incompatible delivery agents can deal with them properly; it queues messages when delivery cannot be done immediately and handles them later; and it recognizes bad addresses and other errors and reroutes or bounces mail as needed.

Here is an example of what was tested successfully at the XTS 300's command line using the Mail program:

Date: 09 Aug 2000 1300 PST

From: bob

To: sue

Subject: Test

This is a test from the command line

Failures were experienced at first due to the fact that input/output mechanisms needed to be modified so that Sendmail could make its necessary communications connections. When Sendmail successfully executed from the command line, and sent the mail message via Mail at the low sensitivity level, problems with the header definitions were discovered. These problems included missing "subject" and "to" lines. The "to" line was easily recovered by modifying the sendmail.cf's header definitions. However, the missing "subject" line was a quirk of the Mail program. The program requires that line be manually typed into the message, along with the "to","cc" and "bcc", etc.

Since the Sendmail program seemed to work from the XTS Terminal at the low sensitivity level, the next step was to test the program from the client machine. Of course, initial errors were encountered. These errors occurred because Sendmail was creating child processes, which the parent processes were unable to communicate with, thus

communications never completed. This was resolved by modifying the svrsmtp.c source file. Pseudosocket communications commands specific to the XTS 300 were added to this file. The source files also required a modification to the flush commands to make them XTS 300 compatible. After the file modifications were made, a successful transmission of a mail message from the client machine's Microsoft Outlook Express program at the low sensitivity level to the XTS 300 was successful. The mail message was retrieved via the Mail program on the XTS 300. However, a problem was encountered with the buffer not being read properly as noted by a missing letter of the SMTP "conversation". The "M" in "Message" was dropped. The problem was fixed by modifying the pseudosocket communications code in the source files. Here is an example of what the audit logs of the XTS 300 displayed for the mail exchange:

bob@here.com... Connecting to sue.there.com. via smtp...

sue.here.com ESMTP Sendmail 8.9.3;Sun, 15 Aug 2000 10:00:25

>>>EHLO golden.rich.com

250 bob.here.com Hello golden.rich [221.185.8.1], pleased to meet you

>>> MAIL From: size=10

250 ...Sender ok

>>>RCPT To:

250 Recipient ok

>>> DATA

354 Enter mail, end with a "." on a line by itself

>>> .

250 WAA12161 essage accepted for delivery

bob@here.com...Sent (WAA12161 Message accepted for delivery)

Closing connection to bob@here.com

>>>QUIT

Once the Microsoft Outlook Express transmission to the XTS 300 was successful, two other client mail programs were tested from the client. Pine, which is a mail program created at the University of Washington, was tested as well as Netscape Communicator. . Early tests of the program brought test failures caused by incorrect configuration within the mail clients themselves. The configuration changes made included ensuring that the right server names

were inserted in the right places, and mail addresses were correct, etc.

Once all three mail clients successfully transmitted to the XTS 300, they were tested for transmission to other clients. Pine was the first mail client software package tested. It succeeded with minimal effort, and the other two programs tested successfully as well. The message transmitted with no errors. Mail was sent and received at the current sensitivity level (SL0-IL3 or sensitivity level unclassified at integrity level 3). Mail transmissions to multiple clients were conducted and the messages were received and read by the addressees at the SL0-IL3 sensitivity/integrity level. The header formats differed between the mail clients due to the different mail client software configurations and various optional fields of the header. However, the following example is typical of the mail format of the mail messages which is in compliance with the RFCs for mail message formats.

Return-Path: here.com

Received: from someplace.here.com with smtp id WA12161 for Sun 15Aug 2000

Message-id:199954564.WAA12162@sue.here.com

X-Sender: bob@someplace.here.com

X-Mailer: PC Pine

Mime-Version:

Content-Type: text/plain

Date:

To:

From:

Subject:

The next hurdle to tackle was testing at higher sensitivity levels. The first test of all of the three mail clients at SL1-IL3, confidential-integrity level 3, failed. Macros in several of the header files, including, the conf.h, sendmail.h, and os_xts.h had to be modified to reorder command calls. Also, modifications to the lock file commands in the conf.c file were made. These final changes ultimately enabled transmission/receipt at all sensitivity/integrity levels across all three mail client platforms.

The final test of Sendmail on the XTS 300 was to test the transmission of attachments via the three client mail

programs. The first preliminary tests failed due to a problem encountered with the IMAP buffer. Sendmail transmitted the attachment and put it on the spool for retrieval by IMAP, but IMAP could not read the attachment off the spool to deliver it to the receiver's mailbox. The problem with the IMAP buffer overflow was corrected, and the client mail programs were able to send and receive attachments without difficulty.

D. DIFFICULTIES

The compiling difficulties encountered with this thesis research were mainly due to the learning curve required for the Sendmail program itself as well as the peculiarities of the STOP 4.4.2 operating system. This system is not widely used and will impede the import of any commercial software to the XTS 300. In addition, programming in the C-language hampered my progress. An extraordinary amount of time to become familiar enough with the Sendmail code in order to make correct changes was required.

IV. CONCLUSIONS AND FUTURE WORK

A. CONCLUSION

Sendmail proved to be a successful mail server platform to run on the XTS 300. All requirements were met:

- It supports a COTS PC client and unmodified COTS mail client software
- It implements the standard SMTP functions as established by RFC 821 and Internet mail headers described in RFC 822.
- Modifications that make the mail server work at multiple security levels did not affect the execution of client mail software such as Microsoft Outlook, Netscape Communicator, and Pine.
- A client operating at a particular sensitivity level was able to receive all mail dominated by that sensitivity level.
- A client operating at a particular sensitivity level was only able to append to and send mail at that particular sensitivity level

The configuration of Sendmail achieved through this research is valid only for local movement of mail (within the MLS LAN). In order for Sendmail to support Internet Mail, additional modification to the configuration file is necessary. In particular, mailer definitions and macros, and additional rewriting rules are needed for addressing and forwarding of mail outside of the MLS LAN as well as

rulesets for how incoming addresses are to be transformed so that the mail reaches the recipient. However, the biggest problem that will be encountered is security. The path between the MLS LAN and the Internet has to somehow be made secure whether it is with cryptographic equipment, Public Key Infrastructure, SSL, or combinations thereof. Then there is the problem of how the mail will be labeled outside of the MLS LAN. Inside the MLS LAN, all mail is labeled at the sensitivity level/integrity level (session level) that the client is in when he or she sends the message. In addition, when a client receives a mail message, it is labeled at the level the sender was in when the message was created. There is currently nothing in place that will instantiate that for mail arriving from outside of the MLS LAN.

B. FUTURE WORK

Additional work is required on the MLS LAN's Sendmail server in order for it to be viable for external mail exchange. The Sendmail configuration file created for the MLS LAN is for local mail delivery only (within the LAN). More work on the configuration file is needed to implement Internet mail exchange into and out of the MLS LAN. Thorough research of secure communications channels and

implementations and designs for the connections of the MLS LAN to the Internet are necessary. In addition, there is currently no way of determining how mail from external clients will be labeled by the XTS 300.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A: MODIFICATIONS TO C-LANGUAGE CODE FOR SOURCE FILES

```
File:  sendmail-8.9.3/src/Build
0a1,542
> #!/bin/sh
>
> # Copyright (c) 1998 Sendmail, Inc.  All rights reserved.
> # Copyright (c) 1993, 1996-1997 Eric P. Allman.  All rights reserved.
> # Copyright (c) 1993
> #   The Regents of the University of California.  All rights reserved.
> #
> # By using this file, you agree to the terms and conditions set
> # forth in the LICENSE file which can be found at the top level of
> # the sendmail distribution.
> #
> #
> #       @(#)Build      8.94 (Berkeley) 1/23/1999
> #
>
> #
> #  A quick-and-dirty script to compile sendmail and related programs
> #  in the presence of multiple architectures.  To use, just use
> #  "sh Build".
> #
>
> #
> # MODIFICATION History:
> # Date      Who                 Comments
> # 01Mar00   David Shifflett     Added XTS-300 specific changes
> #                               Added special link command for XTS-300
> # 08Aug00   David Shifflett     Added creation of the Pseudo-socket
version
> #                               of the makefile
> #
>
> trap "rm -f $obj/.settings$$; exit" 1 2 3 15
>
> # default link command
> LN="ln -s"
>
> cflag=""
> mflag=""
> sflag=""
> makeargs=""
> libdirs=""
> incdirs=""
> libsrch=""
> siteconfig=""
> EX_USAGE=64
> EX_NOINPUT=66
> EX_UNAVAILABLE=69
>
> while [ ! -z "$1" ]
```

```
> do
>     case $1
>     in
>       -c) # clean out existing $obj tree
>           cflag=1
>           shift
>           ;;
>
>       -m) # show Makefile name only
>           mflag=1
>           shift
>           ;;
>
>       -E*)      # environment variables to pass into Build
>           arg=`echo $1 | sed 's/^-E//'`
>           if [ -z "$arg" ]
>           then
>               shift   # move to argument
>               arg=$1
>           fi
>           if [ -z "$arg" ]
>           then
>                 echo "Empty -E flag" >&2
>                 exit $EX_USAGE
>           else
>                 case $arg
>                 in
>                   *=*)      # check format
>                       eval $arg
>                       export `echo $arg | sed 's;=.*;;'`
>                       ;;
>                   *)  # bad format
>                       echo "Bad format for -E argument ($arg)" >&2
>                       exit $EX_USAGE
>                       ;;
>                 esac
>                 shift
>           fi
>           ;;
>
>       -L*)      # set up LIBDIRS
>           libdirs="$libdirs $1"
>           shift
>           ;;
>
>       -I*)      # set up INCDIRS
>           incdirs="$incdirs $1"
>           shift
>           ;;
>
>       -f*)      # select site config file
>           arg=`echo $1 | sed 's/^-f//'`
>           if [ -z "$arg" ]
>           then
>               shift   # move to argument
>               arg=$1
>           fi
```

```
>           if [ "$siteconfig" ]
>           then
>                 echo "Only one -f flag allowed" >&2
>                 exit $EX_USAGE
>           else
>                 siteconfig=$arg
>                 if [ -z "$siteconfig" ]
>                 then
>                       echo "Missing argument for -f flag" >&2
>                       exit $EX_USAGE
>                 elif [ ! -f "$siteconfig" ]
>                 then
>                       echo "${siteconfig}: File not found"
>                       exit $EX_NOINPUT
>                 else
>                       shift # move past argument
>                 fi
>           fi
>           ;;
>
>       -S) # skip auto-configure
>           sflag="-s"
>           shift
>           ;;
>
>       *)  # pass argument to make
>           makeargs="$makeargs \"$1\""
>           shift
>           ;;
>     esac
> done
>
> #
> #  Do heuristic guesses !ONLY! for machines that do not have uname
> #
> if [ -d /NextApps -a ! -f /bin/uname -a ! -f /usr/bin/uname ]
> then
>     # probably a NeXT box
>     arch=`hostinfo | sed -n 's/.*Processor type: \([^ ]*\).*/\1/p'`
>     os=NeXT
>     rel=`hostinfo | sed -n 's/.*NeXT Mach \([0-9\.]*\).*/\1/p'`
> elif [ -f /usr/sony/bin/machine -a -f /etc/osversion ]
> then
>     # probably a Sony NEWS 4.x
>     os=NEWS-OS
>     rel=`awk '{ print $3}' /etc/osversion`
>     arch=`/usr/sony/bin/machine`
> elif [ -d /usr/omron -a -f /bin/luna ]
> then
>     # probably a Omron LUNA
>     os=LUNA
>     if [ -f /bin/luna1 ] && /bin/luna1
>     then
>           rel=unios-b
>           arch=luna1
>     elif [ -f /bin/luna2 ] && /bin/luna2
>     then
```

```
>           rel=Mach
>           arch=luna2
>     elif [ -f /bin/luna88k ] && /bin/luna88k
>     then
>           rel=Mach
>           arch=luna88k
>     fi
> elif [ -d /usr/apollo -a -d \`node_data ]
> then
>     # probably a Apollo/DOMAIN
>     os=DomainOS
>     arch=$ISP
>     rel=`/usr/apollo/bin/bldt | grep Domain | awk '{ print $4 }' | sed
-e 's/,//g'`
> fi
>
> if [ ! "$arch" -a ! "$os" -a ! "$rel" ]
> then
>     arch=`uname -m | sed -e 's/ //g'`
>     os=`uname -s | sed -e 's/\//-/g' -e 's/ //g'`
>     rel=`uname -r | sed -e 's/(/-/g' -e 's/)//g'`
> fi
>
> #
> #  Tweak the values we have already got.  PLEASE LIMIT THESE to
> #  tweaks that are absolutely necessary because your system uname
> #  routine doesn't return something sufficiently unique.  Don't do
> #  it just because you don't like the name that is returned.  You
> #  can combine the architecture name with the os name to create a
> #  unique Makefile name.
> #
>
> # tweak machine architecture
> case $arch
> in
>   sun4*)  arch=sun4;;
>
>   9000/*) arch=`echo $arch | sed -e 's/9000.//' -e 's/..$/xx/'`;;
>
>   DS/907000)    arch=ds90;;
>
>   NILE*)  arch=NILE
>           os=`uname -v`;;
>
>   CRAYT3E|CRAYTS)
>           os=$arch;;
>
> esac
>
> # tweak operating system type and release
> node=`uname -n | sed -e 's/\//-/g' -e 's/ //g'`
> if [ "$os" = "$node" -a "$arch" = "i386" -a "$rel" = 3.2 -a "`uname -
v`" = 2 ]
> then
>     # old versions of SCO UNIX set uname -s the same as uname -n
>     os=SCO_SV
> fi
```

```
>
> if [ "$os" = "$node" -a "$arch" = "i486" -a "`uname -v`" = "STOP" ]
> then
>     # XTS-300 sets uname -s the same as uname -n
>     os=XTS
>     rel=`echo $rel | sed -e 's/\.$//'`
>     # XTS-300 link command, no symbolic links :(
>     LN="ln"
> fi
>
> if [ "$rel" = 4.0 ]
> then
>     case $arch in
>       3[34]??|3[34]??,*)
>           if [ -d /usr/sadm/sysadm/add-ons/WIN-TCP ]
>           then
>                 os=NCR.MP-RAS.2.x
>           elif [ -d /usr/sadm/sysadm/add-ons/inet ]
>           then
>                 os=NCR.MP-RAS.3.x
>           fi
>           ;;
>     esac
> fi
>
> case $os
> in
>   DYNIX-ptx)    os=PTX;;
>   Paragon*)     os=Paragon;;
>   HP-UX)  rel=`echo $rel | sed -e 's/^[^.]*\.0*//'`;;
>   AIX)          rela=$rel
>           rel=`uname -v`
>           case $rel in
>             2)  arch=""
>                 ;;
>             4)  if [ "$rela" = "3" ]
>                 then
>                       arch=$rela
>                 fi
>                 ;;
>           esac
>           rel=$rel.$rela
>           ;;
>   BSD-386)      os=BSD-OS;;
>   SCO_SV) os=SCO; rel=`uname -X | sed -n 's/Release = 3.2v//p'`;;
>   UNIX_System_V) if [ "$arch" = "ds90" ]
>           then
>                 os="UXPDS"
>                 rel=`uname -v | sed -e 's/\(V.*\)L.*/\1/'`
>           fi;;
>   SINIX-?)      os=SINIX;;
>   DomainOS)     case $rel in
>             10.4*)    rel=10.4;;
>           esac
>           ;;
> esac
>
```

```
> # get "base part" of operating system release
> rroot=`echo $rel | sed -e 's/\.[^.]*$//'`
> rbase=`echo $rel | sed -e 's/\..*//'`
> if [ "$rroot" = "$rbase" ]
> then
>     rroot=$rel
> fi
>
> # heuristic tweaks to clean up names -- PLEASE LIMIT THESE!
> if [ "$os" = "unix" ]
> then
>     # might be Altos System V
>     case $rel
>     in
>       5.3*)           os=Altos;;
>     esac
> elif [ -r /unix -a -r /usr/lib/libseq.a -a -r /lib/cpp ]
> then
>     # might be a DYNIX/ptx 2.x system, which has a broken uname
>     if strings /lib/cpp | grep _SEQUENT_ > /dev/null
>     then
>           os=PTX
>     fi
> elif [ -d /usr/nec ]
> then
>     # NEC machine -- what is it running?
>     if [ "$os" = "UNIX_System_V" ]
>     then
>           os=EWS-UX_V
>     elif [ "$os" = "UNIX_SV" ]
>     then
>           os=UX4800
>     fi
> elif [ "$arch" = "mips" ]
> then
>     case $rel
>     in
>       4_*)
>           if [ `uname -v` = "UMIPS" ]
>           then
>                 os=RISCos
>           fi;;
>     esac
> fi
>
> # see if there is a "user suffix" specified
> if [ "${SENDMAIL_SUFFIX-}x" = "x" ]
> then
>     sfx=""
> else
>     sfx=".${SENDMAIL_SUFFIX}"
> fi
>
> echo "Configuration: os=$os, rel=$rel, rbase=$rbase, rroot=$rroot,
arch=$arch, sfx=$sfx"
>
>
```

```
> SMROOT=${SMROOT-..}
> BUILDTOOLS=${BUILDTOOLS-$SMROOT/BuildTools}
> export SMROOT BUILDTOOLS
>
> # see if we are in a Build-able directory
> if [ ! -f Makefile.m4 ]; then
>     echo "Makefile.m4 not found.  Build can only be run from a source
directory."
>     exit $EX_UNAVAILABLE
> fi
>
> # now try to find a reasonable object directory
> if [ -r obj.$os.$rel.$arch$sfx ]; then
>     obj=obj.$os.$rel.$arch$sfx
> elif [ -r obj.$os.$rroot.$arch$sfx ]; then
>     obj=obj.$os.$rroot.$arch$sfx
> elif [ -r obj.$os.$rbase.x.$arch$sfx ]; then
>     obj=obj.$os.$rbase.x.$arch$sfx
> elif [ -r obj.$os.$rel$sfx ]; then
>     obj=obj.$os.$rel$sfx
> elif [ -r obj.$os.$rbase.x$sfx ]; then
>     obj=obj.$os.$rbase.x$sfx
> elif [ -r obj.$os.$arch$sfx ]; then
>     obj=obj.$os.$arch$sfx
> elif [ -r obj.$rel.$arch$sfx ]; then
>     obj=obj.$rel.$arch$sfx
> elif [ -r obj.$rbase.x.$arch$sfx ]; then
>     obj=obj.$rbase.x.$arch$sfx
> elif [ -r obj.$os$sfx ]; then
>     obj=obj.$os$sfx
> elif [ -r obj.$arch$sfx ]; then
>     obj=obj.$arch$sfx
> elif [ -r obj.$rel$sfx ]; then
>     obj=obj.$rel$sfx
> elif [ -r obj$sfx ]; then
>     obj=obj$sfx
> fi
> if [ -z "$obj" -o "$cflag" ]
> then
>     if [ -n "$obj" ]
>     then
>           echo "Clearing out existing $obj tree"
>           rm -rf $obj
>     else
>           # no existing obj directory -- try to create one if Makefile
found
>           obj=obj.$os.$rel.$arch$sfx
>     fi
>     if [ -r $BUILDTOOLS/OS/$os.$rel.$arch$sfx ]; then
>           oscf=$os.$rel.$arch$sfx
>     elif [ -r $BUILDTOOLS/OS/$os.$rel.$arch ]; then
>           oscf=$os.$rel.$arch
>     elif [ -r $BUILDTOOLS/OS/$os.$rroot.$arch$sfx ]; then
>           oscf=$os.$rroot.$arch$sfx
>     elif [ -r $BUILDTOOLS/OS/$os.$rroot.$arch ]; then
>           oscf=$os.$rroot.$arch
>     elif [ -r $BUILDTOOLS/OS/$os.$rbase.x.$arch$sfx ]; then
```

```
>             oscf=$os.$rbase.x.$arch$sfx
>       elif [ -r $BUILDTOOLS/OS/$os.$rbase.x.$arch ]; then
>             oscf=$os.$rbase.x.$arch
>       elif [ -r $BUILDTOOLS/OS/$os.$rel$sfx ]; then
>             oscf=$os.$rel$sfx
>       elif [ -r $BUILDTOOLS/OS/$os.$rel ]; then
>             oscf=$os.$rel
>       elif [ -r $BUILDTOOLS/OS/$os.$rroot$sfx ]; then
>             oscf=$os.$rroot$sfx
>       elif [ -r $BUILDTOOLS/OS/$os.$rroot ]; then
>             oscf=$os.$rroot
>       elif [ -r $BUILDTOOLS/OS/$os.$rbase.x$sfx ]; then
>             oscf=$os.$rbase.x$sfx
>       elif [ -r $BUILDTOOLS/OS/$os.$rbase.x ]; then
>             oscf=$os.$rbase.x
>       elif [ -r $BUILDTOOLS/OS/$os.$arch$sfx ]; then
>             oscf=$os.$arch$sfx
>       elif [ -r $BUILDTOOLS/OS/$os.$arch ]; then
>             oscf=$os.$arch
>       elif [ -r $BUILDTOOLS/OS/$rel.$arch$sfx ]; then
>             oscf=$rel.$arch$sfx
>       elif [ -r $BUILDTOOLS/OS/$rel.$arch ]; then
>             oscf=$rel.$arch
>       elif [ -r $BUILDTOOLS/OS/$rroot.$arch$sfx ]; then
>             oscf=$rroot.$arch$sfx
>       elif [ -r $BUILDTOOLS/OS/$rroot.$arch ]; then
>             oscf=$rroot.$arch
>       elif [ -r $BUILDTOOLS/OS/$rbase.x.$arch$sfx ]; then
>             oscf=$rbase.x.$arch$sfx
>       elif [ -r $BUILDTOOLS/OS/$rbase.x.$arch ]; then
>             oscf=$rbase.x.$arch
>       elif [ -r $BUILDTOOLS/OS/$os$sfx ]; then
>             oscf=$os$sfx
>       elif [ -r $BUILDTOOLS/OS/$os ]; then
>             oscf=$os
>       elif [ -r $BUILDTOOLS/OS/$arch$sfx ]; then
>             oscf=$arch$sfx
>       elif [ -r $BUILDTOOLS/OS/$arch ]; then
>             oscf=$arch
>       elif [ -r $BUILDTOOLS/OS/$rel$sfx ]; then
>             oscf=$rel$sfx
>       elif [ -r $BUILDTOOLS/OS/$rel ]; then
>             oscf=$rel
>       elif [ -r $BUILDTOOLS/OS/$rel$sfx ]; then
>             oscf=$rel$sfx
>       else
>             echo "Cannot determine how to support $arch.$os.$rel" >&2
>             exit $EX_UNAVAILABLE
>       fi
>       M4=`sh $BUILDTOOLS/bin/find_m4.sh`
>       ret=$?
>       if [ $ret -ne 0 ]
>       then
>             exit $ret
>       fi
>       echo "Using M4=$M4"
>       export M4
```

```
>     if [ "$mflag" ]
>     then
>           echo "Will run in virgin $obj using $BUILDTOOLS/OS/$oscf"
>           exit 0
>     fi
>     if [ "$ABI" ]
>     then
>           echo "Using ABI $ABI"
>     fi
>     echo "Creating $obj using $BUILDTOOLS/OS/$oscf"
>     mkdir $obj
>     (cd $obj; $LN ../*.[ch158] .)
>     if [ -f sendmail.hf ]
>     then
>           (cd $obj; $LN ../sendmail.hf .)
>     fi
>
>     rm -f $obj/.settings$$
>     echo 'divert(-1)' > $obj/.settings$$
>     cat $BUILDTOOLS/M4/header.m4 >> $obj/.settings$$
>     if [ "$ABI" ]
>     then
>           echo "define(\`confABI', \`$ABI')" >> $obj/.settings$$
>     fi
>     cat $BUILDTOOLS/OS/$oscf >> $obj/.settings$$
>
>     if [ -z "$siteconfig" ]
>     then
>           # none specified, use defaults
>           if [ -f $BUILDTOOLS/Site/site.$oscf$sfx.m4 ]
>           then
>                 siteconfig=$BUILDTOOLS/Site/site.$oscf$sfx.m4
>           elif [ -f $BUILDTOOLS/Site/site.$oscf.m4 ]
>           then
>                 siteconfig=$BUILDTOOLS/Site/site.$oscf.m4
>           fi
>           if [ -f $BUILDTOOLS/Site/site.config.m4 ]
>           then
>                 siteconfig="$BUILDTOOLS/Site/site.config.m4
$siteconfig"
>           fi
>     fi
>     if [ ! -z "$siteconfig" ]
>     then
>           echo "Including $siteconfig"
>           cat $siteconfig >> $obj/.settings$$
>     fi
>     if [ "$libdirs" ]
>     then
>           echo "define(\`confLIBDIRS', confLIBDIRS \`\`$libdirs'')" >>
$obj/.settings$$
>     fi
>     if [ "$incdirs" ]
>     then
>           echo "define(\`confINCDIRS', confINCDIRS \`\`$incdirs'')" >>
$obj/.settings$$
>     fi
```

```
>     echo 'divert(0)dnl' >> $obj/.settings$$
>     libdirs=`(cat $obj/.settings$$; echo "_SRIDBIL_= confLIBDIRS" ) |
\
>           sed -e 's/\(.\)include/\1_include_/g' -e
's/#define/#_define_/g' | \
>           ${M4} -DconfBUILDTOOLSDIR=$BUILDTOOLS - | \
>           grep "^_SRIDBIL_=" | \
>           sed -e 's/#_define_/#define/g' -e 's/_include_/include/g' -e
"s/^_SRIDBIL_=//"`
>     libsrch=`(cat $obj/.settings$$; echo "_HCRSBIL_= confLIBSEARCH" )
| \
>           sed -e 's/\(.\)include/\1_include_/g' -e
's/#define/#_define_/g' | \
>           ${M4} -DconfBUILDTOOLSDIR=$BUILDTOOLS - | \
>           grep "^_HCRSBIL_=" | \
>           sed -e 's/#_define_/#define/g' -e 's/_include_/include/g' -e
"s/^_HCRSBIL_=//"`
>     echo 'divert(-1)' >> $obj/.settings$$
>     LIBDIRS="$libdirs" LIBSRCH="$libsrch" SITECONFIG="$siteconfig" sh
$BUILDTOOLS/bin/configure.sh $sflag $oscf >> $obj/.settings$$
>     echo 'divert(0)dnl' >> $obj/.settings$$
>     sed -e 's/\(.\)include/\1_include_/g' -e 's/#define/#_define_/g'
$obj/.settings$$ | \
>           ${M4} -DconfBUILDTOOLSDIR=$BUILDTOOLS - Makefile.m4 | \
>           sed -e 's/#_define_/#define/g' -e 's/_include_/include/g' >
$obj/Makefile
>     if [ $? -ne 0 -o ! -s $obj/Makefile ]
>     then
>           echo "ERROR: ${M4} failed; You may need a newer version of
M4, at least as new as System V or GNU" 1>&2
>           rm -rf $obj
>           exit $EX_UNAVAILABLE
>     fi
>     rm -f $obj/.settings$$
>     echo "Making dependencies in $obj"
>     (cd $obj; ${MAKE-make} depend)
>
> # Now Make Pseudo-socket version of Makefile
>     sed -e 's/_XTS/_XTS -DUSE_P_SOCKET/' $obj/Makefile >
$obj/Makefile.pskt
> fi
>
> if [ "$mflag" ]
> then
>     makefile=`ls -l $obj/Makefile | sed 's/.* //'`
>     if [ -z "$makefile" ]
>     then
>           echo "ERROR: $obj exists but has no Makefile" >&2
>           exit $EX_NOINPUT
>     fi
>     echo "Will run in existing $obj using $makefile"
>     exit 0
> fi
>
> echo "Making in $obj"
> cd $obj
> eval exec ${MAKE-make} $makeargs
```

```
File:  sendmail-8.9.3/src/Makefile.m4
6a7,10
> # Modification History
> # Date      Who      Comment
> #-----------------------------------------------------
> # 07Aug00  DJS      Added Pseudo-socket version of executable
106c110,112
< ALL=      sendmail sendmail.st aliases.${MAN5SRC} mailq.${MAN1SRC}
newaliases.${MAN1SRC} sendmail.${MAN8SRC}
---
> # ALL=    sendmail sendmail.st aliases.${MAN5SRC} mailq.${MAN1SRC}
newaliases.${MAN1SRC} sendmail.${MAN8SRC}
> MANALL=   aliases.${MAN5SRC} mailq.${MAN1SRC} newaliases.${MAN1SRC}
sendmail.${MAN8SRC}
> ALL=      sendmail sendmail.st pskt.sendmail
112a119,124
> pskt.sendmail:
>     make -f Makefile.pskt clean_objs sendmail.pskt
>
> sendmail.pskt: ${BEFORE} ${OBJS}
>     ${CC} -o sendmail.pskt ${LDOPTS} ${LIBDIRS} ${OBJS} ${LIBS}
>
148c160,163
<     rm -f ${OBJS} sendmail aliases.${MAN5SRC} mailq.${MAN1SRC}
newaliases.${MAN1SRC} sendmail.${MAN8SRC}
---
>     rm -f ${OBJS} ${ALL}
>
> clean_objs:
>     rm -f ${OBJS}

File: sendmail-8.9.3/src/aliases
1,53c1,2
< #
< #   @(#)aliases 8.2 (Berkeley) 3/5/94
< #
< #  Aliases in this file will NOT be expanded in the header from
< #  Mail, but WILL be visible over networks or from /bin/mail.
< #
< #    >>>>>>>>>>  The program "newaliases" must be run after
< #    >> NOTE >>  this file is updated for any changes to
< #    >>>>>>>>>>  show through to sendmail.
< #
<
< # Basic system aliases -- these MUST be present.
< MAILER-DAEMON:  postmaster
< postmaster:     root
<
< # General redirections for pseudo accounts.
< bin:            root
< daemon:         root
< games:          root
< ingres:         root
< nobody:         root
< system:         root
< toor:           root
```

```
< uucp:           root
<
< # Well-known aliases.
< manager:  root
< dumper:         root
< operator: root
<
< # trap decode to catch security attacks
< decode:         root
<
< # OFFICIAL CSRG/BUG ADDRESSES
<
< # Ftp maintainer.
< ftp: ftp-bugs
< ftp-bugs: bigbug@cs.berkeley.edu
<
< # Distribution office.
< bsd-dist: bsd-dist@cs.berkeley.edu
<
< # Fortune maintainer.
< fortune: fortune@cs.berkeley.edu
<
< # Termcap maintainer.
< termcap: termcap@cs.berkeley.edu
<
< # General bug address.
< ucb-fixes: bigbug@cs.berkeley.edu
< ucb-fixes-request: bigbug@cs.berkeley.edu
< bugs: bugs@cs.berkeley.edu
< # END OFFICIAL BUG ADDRESSES
---
> postmaster: ejbrown
> mailer-daemon: postmaster

File:  sendmail-8.9.3/src/collect.c
10a11,16
>  * Modification History
>  *
>  * Date     Who     Comment
>  * ----------------------------------------------------------
>  * 17Aug00  EJB     Added Pseudo-socket communications commands
>  * 23Aug00  DJS     Added include of os_xts
12a19
>
19a27,33
>
> #ifdef _XTS
> # include "os_xts.h"
> # ifdef USE_P_SOCKET
>   extern int my_fd;               /* Pseudo-socket file descriptor */
> # endif
> #endif
171c185
<                       while (!feof(fp) && !ferror(fp))
---
>                       while (!PFEOF(fp) && !PFERROR(fp))
174c188
```

```
<                               c = getc(fp);
---
>                               c = PBIN(fp);
177c191
<                               clearerr(fp);
---
>                               PCLEARERR(fp);
247c261
<                               ungetc(c, fp);
---
>                               PUNGETC(c, fp);
360c374
<                       clearerr(fp);
---
>                       PCLEARERR(fp);
362c376
<                       c = getc(fp);
---
>                       c = PBIN(fp);
365c379
<                       ungetc(c, fp);
---
>                       PUNGETC(c, fp);
413c427
<     if ((feof(fp) && smtpmode) || ferror(fp))
---
>     if ((PFEOF(fp) && smtpmode) || PFERROR(fp))
451c465
<           if (feof(fp))
---
>           if (PFEOF(fp))
453c467
<           else if (ferror(fp))
---
>           else if (PFERROR(fp))
457c471
<           if (LogLevel > 0 && feof(fp))
---
>           if (LogLevel > 0 && PFEOF(fp))
463c477
<           if (feof(fp))
---
>           if (PFEOF(fp))

File:  sendmail-8.9.3/src/conf.c
16a17,25
> /*
>  * XTS-300 Modifications
>  *
>  * Modification History:
>  * Date     Who               Comments
>  * 01Mar00  David Shifflett   Added (int) typecast to last param of
fcntl()
>  *
>  */
>
966a976
```

```
> printf("RealUid %d\n",RealUid);
968c978,979
<                 if (pw != NULL)
---
>                 if (pw != NULL){
> printf("pw->pw_name %s\n",pw->pw_name);
969a981,983
> }else{
> printf("pw is null\n");
> }
974a989
> printf("kkRealUid %d\n",RealUid);
3442c3457,3458
<     while ((i = fcntl(fd, action, &lfd)) < 0 && errno == EINTR)
---
> #ifndef _XTS
>     while ((i = fcntl(fd, action, (int)&lfd)) < 0 && errno == EINTR)
3443a3460,3477
> #else
>     struct stat statbuf;
>     if(fstat(fd, &statbuf)) {
>        i = 0;
>        printf("not locking(fstat failed)(%s%s, action=%d, type=%d)\n",
>              filename, ext, action, lfd.l_type);
>     }else{
>        if ((statbuf.st_mode & S_IREAD) && (statbuf.st_mode &
S_IWRITE))
>        {
>           while ((i = fcntl(fd, action, (int)&lfd)) < 0 && errno ==
EINTR)
>                 continue;
>        }else{
>           i = 0;
>           printf("not locking(no RW access)(%s%s, action=%d,
type=%d)\n",
>                 filename, ext, action, lfd.l_type);
>        }
>     }
> #endif
3475c3509
<           (void) fcntl(fd, F_GETFL, &omode);
---
>           (void) fcntl(fd, F_GETFL, (int)&omode);
4207a4242
> struct passwd *result;
4213c4248,4255
<     return getpwuid(uid);
---
> printf("uid %d\n",uid);
>     result = getpwuid(uid);
> if (result == NULL) {
> printf("result is null\n");
> }else{
> printf("name is %s\n", result->pw_name);
> }
>     return(result);
```

```
File:  sendmail-8.9.3/src/conf.h
12a13,17
>  *
>  * Modification History
>  * Date     Who     Comment
>  * ----------------------------------------------------------
>  * 23Aug00  DJS     Added XTS defines from os_xts.h
2010a2016,2020
> /* XTS-300 defines */
> #ifdef _XTS
> # define SYSTEM5 1
> #endif /* _XTS */
>
2073a2084
> # ifndef _XTS
2074a2086
> # endif /* _XTS */
2093a2106
> # ifndef _XTS
2096a2110
> # endif /* _XTS */
2098a2113,2164
> /* XTS-300 redefinitions */
> #ifdef _XTS
> # define pid_t int
> # define LOG_EMERG       0       /* system is unusable */
> # define LOG_ALERT       1       /* action must be taken immediately
*/
> # define LOG_CRIT        2       /* critical conditions */
> # define LOG_ERR         3       /* error conditions */
> # define LOG_WARNING     4       /* warning conditions */
> # define LOG_NOTICE      5       /* normal but signification condition
*/
> # define LOG_INFO        6       /* informational */
> # define LOG_DEBUG       7       /* debug-level messages */
> # define LOG_PID         0x01    /*log the pid with each message */
> # define MAXPATHLEN PATH_MAX
> # define HASWAITPID 1
> /* # define HASUNAME 1     included in SYSTEM5 above */
> # define SYS5SIGNALS 1
> # define NOFTRUNCATE 1
> # ifdef HASGETUSERSHELL
> #  undef HASGETUSERSHELL
> # endif
> # define HASGETUSERSHELL 0
> # define NEESFSYNC 1
> # undef HASLSTAT
>
> # ifdef UID_T
> #  undef UID_T
> # endif
> # define UID_T        int
>
> # ifdef _PATH_SENDMAILPID
> #  undef _PATH_SENDMAILPID
> # endif
```

```
> # define _PATH_SENDMAILPID "/usr2/shifflet/wip/sendmail/sendmail-
8.9.3/src/sendmail.pid"
>
> # ifdef _PATH_VENDOR_CF
> #  undef _PATH_VENDOR_CF
> # endif
> # define _PATH_VENDOR_CF "/usr2/shifflet/wip/sendmail/sendmail-
8.9.3/src/sendmail.cf"
>
> # ifndef USE_VENDOR_CF_PATH
> #  define USE_VENDOR_CF_PATH
> # endif
> # ifdef LA_TYPE
> #  undef LA_TYPE
> # endif
> # define LA_TYPE      LA_ZERO
> # ifdef SFS_TYPE
> #  undef SFS_TYPE
> # endif
> # define SFS_TYPE     SFS_NONE
> #endif /* _XTS */
>
2380a2447,2449
> # ifdef SYS_NMLN
> #  undef SYS_NMLN
> # endif
2381a2451
> # ifndef _XTS
2384a2455
> # endif /* ! _XTS */

File:  sendmail-8.9.3/src/daemon.c
10a11,16
>  * Modification History
>  *
>  * Date     Who     Comment
>  * -----------------------------------------------------
>  * 21Aug00  EJB     Added uname because gethostname is broken at
higher levels
>  * 23Aug00  DJS     Added include of sys/select
47a54,57
> #ifdef _XTS
> # include <sys/select.h>
> #endif
>
1209a1220,1222
> #ifdef _XTS
>         struct utsname u_name;
> #endif
1212a1226,1228
>              printf("gethostname failed \n");
>
> #ifndef _XTS
1213a1230,1233
> #else
>                 uname(&u_name);
>           (void) strcpy(hostbuf, u_name.sysname);
```

```
> #endif
1217a1238
>               printf("sm_gethostbyname returns %s %s \n",hp-
>h_name,hostbuf);
1219a1241
>

File:  sendmail-8.9.3/src/deliver.c
10a11,16
>  * Modification History
>  *
>  * Date     Who     Comment
>  * ---------------------------------------------------------
>  * 17Aug00  EJB     Added Pseudo-socket communications commands
>  * 23Aug00  DJS     Added include of os_xts
31a38,44
> #ifdef _XTS
> # include "os_xts.h"
> # ifdef USE_P_SOCKET
>   extern int my_fd;                /* Pseudo-socket file descriptor */
> # endif
> #endif
>
1612c1625
<           (void) fflush(stdout);
---
>           (void) PFLUSH(stdout);
3588c3601
<           (void) fflush(stdout);
---
>           (void) PFLUSH(stdout);

file:  sendmail-8.9.3/src/err.c
10a11,16
>  * Modification History
>  *
>  * Date     Who     Comment
>  * ---------------------------------------------------------
>  * 17Aug00  EJB     Added Pseudo-socket communications commands
>  * 23Aug00  DJS     Added include of os_xts
19a26,34
> #ifdef _XTS
> extern FILE *tmpf;
> extern char *tmpf_out;
> #include "os_xts.h"
> #ifdef USE_P_SOCKET
> extern int my_fd;                /* Pseudo-socket file descriptor */
> #endif
> #endif // _XTS
>
366a382,390
> #ifdef _XTS
>    printf("putoutmsg [%s]\n", msg);
>    printf("holdmsg   [%s]\n", (holdmsg?"TRUE":"FALSE"));
>    printf("heldmsg   [%s]\n", (heldmsg?"TRUE":"FALSE"));
>    // TMP_OUT_1(tmpf_out, "putoutmsg [%s]\n", msg, tmpf);
```

```
>    // TMP_OUT_1(tmpf_out, "holdmsg    [%s]\n",
(holdmsg?"TRUE":"FALSE"), tmpf);
>    // TMP_OUT_1(tmpf_out, "heldmsg    [%s]\n",
(heldmsg?"TRUE":"FALSE"), tmpf);
> #endif // _XTS
>
402c426
<     (void) fflush(stdout);
---
>     (void) PFLUSH(stdout);
409a434,435
>         {
> #ifndef USE_P_SOCKET
411c437,443
<     else
---
> #else
>           PSOUT(msg, OutChannel);
>           PBOUT('\r');
>           PBOUT('\n');
> #endif // USE_P_SOCKET
>     }else{
> #ifndef USE_P_SOCKET
412a445,449
> #else
>           PSOUT(&msg[4], OutChannel);
>           PBOUT('\n');
> #endif // USE_P_SOCKET
>         }
417c454
<           (void) fflush(OutChannel);
---
>           (void) PFLUSH(OutChannel);

File:  sendmail-8.9.3/src/main.c
10a11,17
>  * Modification History
>  *
>  * Date     Who     Comment
>  * -----------------------------------------------------
>  * 17Aug00  EJB     Added Pseudo-socket communications commands
>  * 17Aug00  DJS     Handle EINTR in PSKT input, enable debugging
>  * 23Aug00  DJS     Added include of os_xts
75a83,93
> /* XTS-300 Debugging variables */
> #ifdef _XTS
> FILE *tmpf = NULL;                /* DJS - debugging file descriptor */
> char tmpf_out[1000];
> char *myHomeDir = NULL;           /* DJS - used by openlog */
> #include "os_xts.h"
> #ifdef USE_P_SOCKET
> int my_fd;                        /* Pseudo-socket file descriptor */
> #endif
> #endif // _XTS
>
92a111,192
> #ifdef USE_P_SOCKET
```

```
>     // internal functions to handle read with timeout
>     void handler(int signo)
>     {
>       // do nothing, just return
>       int debug_on = 1;
>       dbugd(debug_on, "handler(): entered signo = " , signo);
>     }
>
>     /* Wait for input to be available in the PSKT
>      * Accepts: timeout in seconds
>      * Returns: 1 if have input, else 0
>      */
>
>     long server_input_wait (long seconds)
>     {
>       long result;
>       int sel_res;
>       fd_set rfd, xfd;
>       struct timeval tmo;
>       tmo.tv_sec = seconds; tmo.tv_usec = 0;
>       FD_ZERO (&rfd);
>       FD_ZERO (&xfd);
>       FD_SET (my_fd,&rfd);
>       FD_SET (my_fd,&xfd);
>       sel_res = pskt_select_cli(my_fd+1,&rfd,0,&xfd,&tmo);
>       if (FD_ISSET(my_fd,&xfd))
>       {
>          result = -1;
>       }else if (FD_ISSET(my_fd,&rfd))
>       {
>          result = sel_res ? 1 : 0;
>       }else{
>          result = 0;
>       }
>       return result;
>     }
>
>     int psin_with_pause(char *s, int length)
>     {
>        // Do a 'zero' delay select,
>        // just in case data is waiting there
>        // and we have already gotten the signal
>        // from the SSS child process
>        long wait_res = server_input_wait(0);
>
>        if (!wait_res)
>        {
>       // Nothing waiting, pause until data is available
>       pause();
>        }
>        if (wait_res == -1)
>        {
>       return(0);
>        }else{
>       if (errno == EINTR) errno = 0;
>       return(pskt_read_stop_at_cli(my_fd, '\n', s, length));
>        }
```

```
>     }
>     int pbin_with_pause()
>     {
>        // Do a 'zero' delay select,
>        // just in case data is waiting there
>        // and we have already gotten the signal
>        // from the SSS child process
>        long wait_res = server_input_wait(0);
>
>        if (!wait_res)
>        {
>       // Nothing waiting, pause until data is available
>       pause();
>        }
>        if (wait_res == -1)
>        {
>       return(0);
>        }else{
>       if (errno == EINTR) errno = 0;
>           return(pskt_read_char_cli(my_fd));
>        }
>     }
> #endif
>
149a250,258
> #ifdef _XTS
>     sprintf (tmpf_out,"/tmp/smtp_%d.tmp",getpid());
>     tmpf = fopen (tmpf_out,"a+");
>     TMP_OUT_0("hello world\n", tmpf);
>     TMP_OUT_1(tmpf_out, "sendmail started [%d]\n", getpid(), tmpf);
> #endif
> #ifdef USE_P_SOCKET
>     sprintf (tmpf_out,"/tmp/smtp_%d.tmp",getpid());
>     FILE *tmpf2 = freopen (tmpf_out,"a",stdout);
150a260,316
>     int shmid, debug_on = 1;
>     int result, pskt_handle;
>     access_ma my_sess_level;
>     get_current_level(&my_sess_level);
>
>     // Find our PSKT, using the PSKT Map DB
>     result = access_pmap_db();
>     if (result == PMAP_INITIALIZED)
>     {
>        struct passwd *pw;
>        unsigned long euid;
>        euid = geteuid ();
>        if (pw = getpwuid (euid))
>        {
>       result = get_pskt_handle(pw->pw_name, my_sess_level,
&pskt_handle);
>        }else{
>       TMP_OUT_0("Could NOT acess pw entry\n", tmpf);
>       exit(1);
>        }
>     }else{
>        TMP_OUT_0("Could NOT access PMAP DB\n", tmpf);
```

```
>       exit(1);
>    }
>
>    if (result == PMAP_FOUND)
>    {
>       TMP_OUT_1(tmpf_out,"Using PSKT [%d]\n", pskt_handle, tmpf);
>    }else{
>       TMP_OUT_0("Could NOT find PSKT\n", tmpf);
>       exit(1);
>    }
>
>    // initialize access to the PSKT
>    result = pskt_attach(pskt_handle);
>    if (result != PSKT_INITIALIZED)
>    {
>       TMP_OUT_0("pskt_attach error\n", tmpf);
>       exit(1);
>    }
>
>    // find the PSKT connection we are supposed to use
>    result = pskt_find_connection(getpid(), &my_fd);
>    if (result != PSKT_FOUND)
>    {
>       TMP_OUT_0("pskt_find_connection error\n", tmpf);
>       exit(1);
>    }else{
>       TMP_OUT_1(tmpf_out,"Using PSKT fd [%d]\n", my_fd, tmpf);
>    }
>
>    // make sure we handle signals from the SSS child process
>    sigset(SIGURG, handler);
>
>    // make sure we speed up data transfer with flush calls
>    pskt_flush_required();
> #endif
>
277a444,447
> #if defined(_XTS) && defined(USE_P_SOCKET)
>         // uncomment and modify the next line to enable debugging
>     tTflag("55.60");
> #endif
297a468,469
> printf("RealUid [%d]\n", RealUid);
> // TMP_OUT_1(tmpf_out,"RealUid [%d]\n", RealUid, tmpf);
350a523,524
> printf("Done saving args\n");
> // TMP_OUT_0("Done saving args\n", tmpf);
364c538
<                       putchar('\n');
---
>                       PBOUT('\n');
369,370c543,544
<                       putchar('\t');
<                       putchar('\t');
---
>                       PBOUT('\t');
>                       PBOUT('\t');
```

```
373c547
<                       putchar(' ');
---
>                       PBOUT(' ');
377c551
<           putchar('\n');
---
>           PBOUT('\n');
391c565
<                       putchar('\n');
---
>                       PBOUT('\n');
396,397c570,571
<                       putchar('\t');
<                       putchar('\t');
---
>                       PBOUT('\t');
>                       PBOUT('\t');
400c574
<                       putchar(' ');
---
>                       PBOUT(' ');
404c578
<           putchar('\n');
---
>           PBOUT('\n');
414a589,593
> printf("Set InChannel [%d]\n", InChannel);
> printf("Set OutChannel [%d]\n", OutChannel);
> // TMP_OUT_1(tmpf_out,"Set InChannel [%d]\n", InChannel, tmpf);
> // TMP_OUT_1(tmpf_out,"Set OutChannel [%d]\n", OutChannel, tmpf);
>
571a751,752
> printf("arg option j [%c]\n", j);
> // TMP_OUT_1(tmpf_out,"arg option j [%c]\n", j, tmpf);
611a793,794
> printf("2nd arg j [%c]\n", j);
> // TMP_OUT_1(tmpf_out,"2nd arg j [%c]\n", j, tmpf);
842c1025,1030
<
---
> #if SMTP && USE_P_SOCKET
> // Hard-coded to SMTP option
> OpMode = MD_SMTP;
> #endif
> printf("Done args OpMode [%c]\n", OpMode);
> // TMP_OUT_1(tmpf_out,"Done args OpMode [%c]\n", OpMode, tmpf);
1384,1385c1572,1573
<                   (void) fflush(stdout);
<                   if (fgets(buf, sizeof buf, stdin) == NULL)
---
>                   (void) PFLUSH(stdout);
>                   if (PSIN(buf, sizeof buf, stdin) == NULL)
1487a1676,1677
> printf("In SMTP ifdef\n");
> // TMP_OUT_0("In SMTP ifdef\n", tmpf);
1936c2126
```

```
<           (void) fflush(stdout);
---
>           (void) PFLUSH(stdout);
2429c2619
<                       putchar('R');
---
>                       PBOUT('R');
2434c2624
<                           putchar(' ');
---
>                           PBOUT(' ');
2436,2437c2626,2627
<                       putchar('\t');
<                       putchar('\t');
---
>                       PBOUT('\t');
>                       PBOUT('\t');
2442c2632
<                           putchar(' ');
---
>                           PBOUT(' ');
2444c2634
<                       putchar('\n');
---
>                       PBOUT('\n');

File:  sendmail-8.9.3/src/makesendmail
0a1,542
> #!/bin/sh
>
> # Copyright (c) 1998 Sendmail, Inc.  All rights reserved.
> # Copyright (c) 1993, 1996-1997 Eric P. Allman.  All rights reserved.
> # Copyright (c) 1993
> #   The Regents of the University of California.  All rights reserved.
> #
> # By using this file, you agree to the terms and conditions set
> # forth in the LICENSE file which can be found at the top level of
> # the sendmail distribution.
> #
> #
> #       @(#)Build     8.94 (Berkeley) 1/23/1999
> #
>
> #
> #  A quick-and-dirty script to compile sendmail and related programs
> #  in the presence of multiple architectures.  To use, just use
> #  "sh Build".
> #
>
> #
> # MODIFICATION History:
> # Date     Who                Comments
> # 01Mar00  David Shifflett    Added XTS-300 specific changes
> #                             Added special link command for XTS-300
> # 08Aug00  David Shifflett    Added creation of the Pseudo-socket
version
> #                             of the makefile
```

```
> #
>
> trap "rm -f $obj/.settings$$; exit" 1 2 3 15
>
> # default link command
> LN="ln -s"
>
> cflag=""
> mflag=""
> sflag=""
> makeargs=""
> libdirs=""
> incdirs=""
> libsrch=""
> siteconfig=""
> EX_USAGE=64
> EX_NOINPUT=66
> EX_UNAVAILABLE=69
>
> while [ ! -z "$1" ]
> do
>     case $1
>     in
>       -c) # clean out existing $obj tree
>           cflag=1
>           shift
>           ;;
>
>       -m) # show Makefile name only
>           mflag=1
>           shift
>           ;;
>
>       -E*)      # environment variables to pass into Build
>           arg=`echo $1 | sed 's/^-E//'`
>           if [ -z "$arg" ]
>           then
>               shift   # move to argument
>               arg=$1
>           fi
>           if [ -z "$arg" ]
>           then
>                 echo "Empty -E flag" >&2
>                 exit $EX_USAGE
>           else
>                 case $arg
>                 in
>                   *=*)      # check format
>                       eval $arg
>                       export `echo $arg | sed 's;=.*;;'`
>                       ;;
>                   *)  # bad format
>                       echo "Bad format for -E argument ($arg)" >&2
>                       exit $EX_USAGE
>                       ;;
>                 esac
>                 shift
```

```
>            fi
>            ;;
>
>        -L*)        # set up LIBDIRS
>            libdirs="$libdirs $1"
>            shift
>            ;;
>
>        -I*)        # set up INCDIRS
>            incdirs="$incdirs $1"
>            shift
>            ;;
>
>        -f*)        # select site config file
>            arg=`echo $1 | sed 's/^-f//'`
>            if [ -z "$arg" ]
>            then
>                shift   # move to argument
>                arg=$1
>            fi
>            if [ "$siteconfig" ]
>            then
>                  echo "Only one -f flag allowed" >&2
>                  exit $EX_USAGE
>            else
>                  siteconfig=$arg
>                  if [ -z "$siteconfig" ]
>                  then
>                        echo "Missing argument for -f flag" >&2
>                        exit $EX_USAGE
>                  elif [ ! -f "$siteconfig" ]
>                  then
>                        echo "${siteconfig}: File not found"
>                        exit $EX_NOINPUT
>                  else
>                        shift # move past argument
>                  fi
>            fi
>            ;;
>
>        -S) # skip auto-configure
>            sflag="-s"
>            shift
>            ;;
>
>        *)  # pass argument to make
>            makeargs="$makeargs \"$1\""
>            shift
>            ;;
>     esac
> done
>
> #
> #  Do heuristic guesses !ONLY! for machines that do not have uname
> #
> if [ -d /NextApps -a ! -f /bin/uname -a ! -f /usr/bin/uname ]
> then
```

```
>     # probably a NeXT box
>     arch=`hostinfo | sed -n 's/.*Processor type: \([^ ]*\).*/\1/p'`
>     os=NeXT
>     rel=`hostinfo | sed -n 's/.*NeXT Mach \([0-9\.]*\).*/\1/p'`
> elif [ -f /usr/sony/bin/machine -a -f /etc/osversion ]
> then
>     # probably a Sony NEWS 4.x
>     os=NEWS-OS
>     rel=`awk '{ print $3}' /etc/osversion`
>     arch=`/usr/sony/bin/machine`
> elif [ -d /usr/omron -a -f /bin/luna ]
> then
>     # probably a Omron LUNA
>     os=LUNA
>     if [ -f /bin/luna1 ] && /bin/luna1
>     then
>           rel=unios-b
>           arch=luna1
>     elif [ -f /bin/luna2 ] && /bin/luna2
>     then
>           rel=Mach
>           arch=luna2
>     elif [ -f /bin/luna88k ] && /bin/luna88k
>     then
>           rel=Mach
>           arch=luna88k
>     fi
> elif [ -d /usr/apollo -a -d \`node_data ]
> then
>     # probably a Apollo/DOMAIN
>     os=DomainOS
>     arch=$ISP
>     rel=`/usr/apollo/bin/bldt | grep Domain | awk '{ print $4 }' | sed
-e 's/,//g'`
> fi
>
> if [ ! "$arch" -a ! "$os" -a ! "$rel" ]
> then
>     arch=`uname -m | sed -e 's/ //g'`
>     os=`uname -s | sed -e 's/\//-/g' -e 's/ //g'`
>     rel=`uname -r | sed -e 's/(/-/g' -e 's/)//g'`
> fi
>
> #
> #  Tweak the values we have already got.  PLEASE LIMIT THESE to
> #  tweaks that are absolutely necessary because your system uname
> #  routine doesn't return something sufficiently unique.  Don't do
> #  it just because you don't like the name that is returned.  You
> #  can combine the architecture name with the os name to create a
> #  unique Makefile name.
> #
>
> # tweak machine architecture
> case $arch
> in
>   sun4*)  arch=sun4;;
>
```

```
>   9000/*) arch=`echo $arch | sed -e 's/9000.//' -e 's/..$/xx/'`;;
>
>   DS/907000)    arch=ds90;;
>
>   NILE*)  arch=NILE
>           os=`uname -v`;;
>
>   CRAYT3E|CRAYTS)
>           os=$arch;;
>
> esac
>
> # tweak operating system type and release
> node=`uname -n | sed -e 's/\//-/g' -e 's/ //g'`
> if [ "$os" = "$node" -a "$arch" = "i386" -a "$rel" = 3.2 -a "`uname -
v`" = 2 ]
> then
>     # old versions of SCO UNIX set uname -s the same as uname -n
>     os=SCO_SV
> fi
>
> if [ "$os" = "$node" -a "$arch" = "i486" -a "`uname -v`" = "STOP" ]
> then
>     # XTS-300 sets uname -s the same as uname -n
>     os=XTS
>     rel=`echo $rel | sed -e 's/\.$//'`
>     # XTS-300 link command, no symbolic links :(
>     LN="ln"
> fi
>
> if [ "$rel" = 4.0 ]
> then
>     case $arch in
>       3[34]??|3[34]??,*)
>           if [ -d /usr/sadm/sysadm/add-ons/WIN-TCP ]
>           then
>                 os=NCR.MP-RAS.2.x
>           elif [ -d /usr/sadm/sysadm/add-ons/inet ]
>           then
>                 os=NCR.MP-RAS.3.x
>           fi
>           ;;
>     esac
> fi
>
> case $os
> in
>   DYNIX-ptx)    os=PTX;;
>   Paragon*)     os=Paragon;;
>   HP-UX)  rel=`echo $rel | sed -e 's/^[^.]*\.0*//'`;;
>   AIX)          rela=$rel
>           rel=`uname -v`
>           case $rel in
>             2)  arch=""
>                 ;;
>             4)  if [ "$rela" = "3" ]
>                 then
```

```
>                       arch=$rela
>                 fi
>                 ;;
>           esac
>           rel=$rel.$rela
>           ;;
>   BSD-386)      os=BSD-OS;;
>   SCO_SV) os=SCO; rel=`uname -X | sed -n 's/Release = 3.2v//p'`;;
>   UNIX_System_V) if [ "$arch" = "ds90" ]
>           then
>                 os="UXPDS"
>                 rel=`uname -v | sed -e 's/\(V.*\)L.*/\1/'`
>           fi;;
>   SINIX-?)      os=SINIX;;
>   DomainOS)     case $rel in
>             10.4*)    rel=10.4;;
>           esac
>           ;;
> esac
>
> # get "base part" of operating system release
> rroot=`echo $rel | sed -e 's/\.[^.]*$//'`
> rbase=`echo $rel | sed -e 's/\..*//'`
> if [ "$rroot" = "$rbase" ]
> then
>     rroot=$rel
> fi
>
> # heuristic tweaks to clean up names -- PLEASE LIMIT THESE!
> if [ "$os" = "unix" ]
> then
>     # might be Altos System V
>     case $rel
>     in
>       5.3*)           os=Altos;;
>     esac
> elif [ -r /unix -a -r /usr/lib/libseq.a -a -r /lib/cpp ]
> then
>     # might be a DYNIX/ptx 2.x system, which has a broken uname
>     if strings /lib/cpp | grep _SEQUENT_ > /dev/null
>     then
>           os=PTX
>     fi
> elif [ -d /usr/nec ]
> then
>     # NEC machine -- what is it running?
>     if [ "$os" = "UNIX_System_V" ]
>     then
>           os=EWS-UX_V
>     elif [ "$os" = "UNIX_SV" ]
>     then
>           os=UX4800
>     fi
> elif [ "$arch" = "mips" ]
> then
>     case $rel
>     in
```

```
>       4_*)
>           if [ `uname -v` = "UMIPS" ]
>           then
>                 os=RISCos
>           fi;;
>     esac
> fi
>
> # see if there is a "user suffix" specified
> if [ "${SENDMAIL_SUFFIX-}x" = "x" ]
> then
>     sfx=""
> else
>     sfx=".${SENDMAIL_SUFFIX}"
> fi
>
> echo "Configuration: os=$os, rel=$rel, rbase=$rbase, rroot=$rroot,
arch=$arch, sfx=$sfx"
>
>
> SMROOT=${SMROOT-..}
> BUILDTOOLS=${BUILDTOOLS-$SMROOT/BuildTools}
> export SMROOT BUILDTOOLS
>
> # see if we are in a Build-able directory
> if [ ! -f Makefile.m4 ]; then
>     echo "Makefile.m4 not found.  Build can only be run from a source
directory."
>     exit $EX_UNAVAILABLE
> fi
>
> # now try to find a reasonable object directory
> if [ -r obj.$os.$rel.$arch$sfx ]; then
>     obj=obj.$os.$rel.$arch$sfx
> elif [ -r obj.$os.$rroot.$arch$sfx ]; then
>     obj=obj.$os.$rroot.$arch$sfx
> elif [ -r obj.$os.$rbase.x.$arch$sfx ]; then
>     obj=obj.$os.$rbase.x.$arch$sfx
> elif [ -r obj.$os.$rel$sfx ]; then
>     obj=obj.$os.$rel$sfx
> elif [ -r obj.$os.$rbase.x$sfx ]; then
>     obj=obj.$os.$rbase.x$sfx
> elif [ -r obj.$os.$arch$sfx ]; then
>     obj=obj.$os.$arch$sfx
> elif [ -r obj.$rel.$arch$sfx ]; then
>     obj=obj.$rel.$arch$sfx
> elif [ -r obj.$rbase.x.$arch$sfx ]; then
>     obj=obj.$rbase.x.$arch$sfx
> elif [ -r obj.$os$sfx ]; then
>     obj=obj.$os$sfx
> elif [ -r obj.$arch$sfx ]; then
>     obj=obj.$arch$sfx
> elif [ -r obj.$rel$sfx ]; then
>     obj=obj.$rel$sfx
> elif [ -r obj$sfx ]; then
>     obj=obj$sfx
> fi
```

```
> if [ -z "$obj" -o "$cflag" ]
> then
>     if [ -n "$obj" ]
>     then
>          echo "Clearing out existing $obj tree"
>          rm -rf $obj
>     else
>          # no existing obj directory -- try to create one if Makefile
found
>          obj=obj.$os.$rel.$arch$sfx
>     fi
>     if [ -r $BUILDTOOLS/OS/$os.$rel.$arch$sfx ]; then
>          oscf=$os.$rel.$arch$sfx
>     elif [ -r $BUILDTOOLS/OS/$os.$rel.$arch ]; then
>          oscf=$os.$rel.$arch
>     elif [ -r $BUILDTOOLS/OS/$os.$rroot.$arch$sfx ]; then
>          oscf=$os.$rroot.$arch$sfx
>     elif [ -r $BUILDTOOLS/OS/$os.$rroot.$arch ]; then
>          oscf=$os.$rroot.$arch
>     elif [ -r $BUILDTOOLS/OS/$os.$rbase.x.$arch$sfx ]; then
>          oscf=$os.$rbase.x.$arch$sfx
>     elif [ -r $BUILDTOOLS/OS/$os.$rbase.x.$arch ]; then
>          oscf=$os.$rbase.x.$arch
>     elif [ -r $BUILDTOOLS/OS/$os.$rel$sfx ]; then
>          oscf=$os.$rel$sfx
>     elif [ -r $BUILDTOOLS/OS/$os.$rel ]; then
>          oscf=$os.$rel
>     elif [ -r $BUILDTOOLS/OS/$os.$rroot$sfx ]; then
>          oscf=$os.$rroot$sfx
>     elif [ -r $BUILDTOOLS/OS/$os.$rroot ]; then
>          oscf=$os.$rroot
>     elif [ -r $BUILDTOOLS/OS/$os.$rbase.x$sfx ]; then
>          oscf=$os.$rbase.x$sfx
>     elif [ -r $BUILDTOOLS/OS/$os.$rbase.x ]; then
>          oscf=$os.$rbase.x
>     elif [ -r $BUILDTOOLS/OS/$os.$arch$sfx ]; then
>          oscf=$os.$arch$sfx
>     elif [ -r $BUILDTOOLS/OS/$os.$arch ]; then
>          oscf=$os.$arch
>     elif [ -r $BUILDTOOLS/OS/$rel.$arch$sfx ]; then
>          oscf=$rel.$arch$sfx
>     elif [ -r $BUILDTOOLS/OS/$rel.$arch ]; then
>          oscf=$rel.$arch
>     elif [ -r $BUILDTOOLS/OS/$rroot.$arch$sfx ]; then
>          oscf=$rroot.$arch$sfx
>     elif [ -r $BUILDTOOLS/OS/$rroot.$arch ]; then
>          oscf=$rroot.$arch
>     elif [ -r $BUILDTOOLS/OS/$rbase.x.$arch$sfx ]; then
>          oscf=$rbase.x.$arch$sfx
>     elif [ -r $BUILDTOOLS/OS/$rbase.x.$arch ]; then
>          oscf=$rbase.x.$arch
>     elif [ -r $BUILDTOOLS/OS/$os$sfx ]; then
>          oscf=$os$sfx
>     elif [ -r $BUILDTOOLS/OS/$os ]; then
>          oscf=$os
>     elif [ -r $BUILDTOOLS/OS/$arch$sfx ]; then
>          oscf=$arch$sfx
```

```
>       elif [ -r $BUILDTOOLS/OS/$arch ]; then
>             oscf=$arch
>       elif [ -r $BUILDTOOLS/OS/$rel$sfx ]; then
>             oscf=$rel$sfx
>       elif [ -r $BUILDTOOLS/OS/$rel ]; then
>             oscf=$rel
>       elif [ -r $BUILDTOOLS/OS/$rel$sfx ]; then
>             oscf=$rel$sfx
>       else
>             echo "Cannot determine how to support $arch.$os.$rel" >&2
>             exit $EX_UNAVAILABLE
>       fi
>       M4=`sh $BUILDTOOLS/bin/find_m4.sh`
>       ret=$?
>       if [ $ret -ne 0 ]
>       then
>             exit $ret
>       fi
>       echo "Using M4=$M4"
>       export M4
>       if [ "$mflag" ]
>       then
>             echo "Will run in virgin $obj using $BUILDTOOLS/OS/$oscf"
>             exit 0
>       fi
>       if [ "$ABI" ]
>       then
>             echo "Using ABI $ABI"
>       fi
>       echo "Creating $obj using $BUILDTOOLS/OS/$oscf"
>       mkdir $obj
>       (cd $obj; $LN ../*.[ch158] .)
>       if [ -f sendmail.hf ]
>       then
>             (cd $obj; $LN ../sendmail.hf .)
>       fi
>
>       rm -f $obj/.settings$$
>       echo 'divert(-1)' > $obj/.settings$$
>       cat $BUILDTOOLS/M4/header.m4 >> $obj/.settings$$
>       if [ "$ABI" ]
>       then
>             echo "define(\`confABI', \`$ABI')" >> $obj/.settings$$
>       fi
>       cat $BUILDTOOLS/OS/$oscf >> $obj/.settings$$
>
>       if [ -z "$siteconfig" ]
>       then
>             # none specified, use defaults
>             if [ -f $BUILDTOOLS/Site/site.$oscf$sfx.m4 ]
>             then
>                   siteconfig=$BUILDTOOLS/Site/site.$oscf$sfx.m4
>             elif [ -f $BUILDTOOLS/Site/site.$oscf.m4 ]
>             then
>                   siteconfig=$BUILDTOOLS/Site/site.$oscf.m4
>             fi
>             if [ -f $BUILDTOOLS/Site/site.config.m4 ]
```

```
>           then
>                 siteconfig="$BUILDTOOLS/Site/site.config.m4
$siteconfig"
>           fi
>     fi
>     if [ ! -z "$siteconfig" ]
>     then
>           echo "Including $siteconfig"
>           cat $siteconfig >> $obj/.settings$$
>     fi
>     if [ "$libdirs" ]
>     then
>           echo "define(\`confLIBDIRS', confLIBDIRS \`\`$libdirs'')" >>
$obj/.settings$$
>     fi
>     if [ "$incdirs" ]
>     then
>           echo "define(\`confINCDIRS', confINCDIRS \`\`$incdirs'')" >>
$obj/.settings$$
>     fi
>     echo 'divert(0)dnl' >> $obj/.settings$$
>     libdirs=`(cat $obj/.settings$$; echo "_SRIDBIL_= confLIBDIRS" ) |
\
>           sed -e 's/\(.\)include/\1_include_/g' -e
's/#define/#_define_/g' | \
>           ${M4} -DconfBUILDTOOLSDIR=$BUILDTOOLS - | \
>           grep "^_SRIDBIL_=" | \
>           sed -e 's/#_define_/#define/g' -e 's/_include_/include/g' -e
"s/^_SRIDBIL_=//"`
>     libsrch=`(cat $obj/.settings$$; echo "_HCRSBIL_= confLIBSEARCH" )
| \
>           sed -e 's/\(.\)include/\1_include_/g' -e
's/#define/#_define_/g' | \
>           ${M4} -DconfBUILDTOOLSDIR=$BUILDTOOLS - | \
>           grep "^_HCRSBIL_=" | \
>           sed -e 's/#_define_/#define/g' -e 's/_include_/include/g' -e
"s/^_HCRSBIL_=//"`
>     echo 'divert(-1)' >> $obj/.settings$$
>     LIBDIRS="$libdirs" LIBSRCH="$libsrch" SITECONFIG="$siteconfig" sh
$BUILDTOOLS/bin/configure.sh $sflag $oscf >> $obj/.settings$$
>     echo 'divert(0)dnl' >> $obj/.settings$$
>     sed -e 's/\(.\)include/\1_include_/g' -e 's/#define/#_define_/g'
$obj/.settings$$ | \
>           ${M4} -DconfBUILDTOOLSDIR=$BUILDTOOLS - Makefile.m4 | \
>           sed -e 's/#_define_/#define/g' -e 's/_include_/include/g' >
$obj/Makefile
>     if [ $? -ne 0 -o ! -s $obj/Makefile ]
>     then
>           echo "ERROR: ${M4} failed; You may need a newer version of
M4, at least as new as System V or GNU" 1>&2
>           rm -rf $obj
>           exit $EX_UNAVAILABLE
>     fi
>     rm -f $obj/.settings$$
>     echo "Making dependencies in $obj"
>     (cd $obj; ${MAKE-make} depend)
>
```

```
> # Now Make Pseudo-socket version of Makefile
>     sed -e 's/_XTS/_XTS -DUSE_P_SOCKET/' $obj/Makefile >
$obj/Makefile.pskt
> fi
>
> if [ "$mflag" ]
> then
>     makefile=`ls -l $obj/Makefile | sed 's/.* //'`
>     if [ -z "$makefile" ]
>     then
>           echo "ERROR: $obj exists but has no Makefile" >&2
>           exit $EX_NOINPUT
>     fi
>     echo "Will run in existing $obj using $makefile"
>     exit 0
> fi
>
> echo "Making in $obj"
> cd $obj
> eval exec ${MAKE-make} $makeargs

File:  sendmail-8.9.3/src/parseaddr.c
10a11,16
>  * Modification History
>  *
>  * Date     Who     Comment
>  * ----------------------------------------------------
>  * 17Aug00  EJB     Added Pseudo-socket communications commands
>  * 23Aug00  DJS     Added include of os_xts
18a25,31
> #ifdef _XTS
> # include "os_xts.h"
> # ifdef USE_P_SOCKET
>   extern int my_fd;                /* Pseudo-socket file descriptor */
> # endif
> #endif
>
473c486
<           (void) putchar('\n');
---
>           (void) PBOUT('\n');
644c657
<                       (void) putchar('\n');
---
>                       (void) PBOUT('\n');
1032c1045
<                                 (void) fflush(stdout);
---
>                                 (void) PFLUSH(stdout);
1947c1960
<           (void) fflush(stdout);
---
>           (void) PFLUSH(stdout);

File:  sendmail-8.9.3/src/readcf.c
10a11,16
>  * Modification History
```

```
>  *
>  * Date     Who     Comment
>  * ----------------------------------------------------
>  * 17Aug00  EJB     Added Pseudo-socket communications commands
>  * 23Aug00  DJS     Added include of os_xts
22a29,35
> #ifdef _XTS
> # include "os_xts.h"
> # ifdef USE_P_SOCKET
>   extern int my_fd;                /* Pseudo-socket file descriptor */
> # endif
> #endif
>
1307c1320
<                 (void) putchar(j);
---
>                 (void) PBOUT(j);

File:  sendmail-8.9.3/src/savemail.c
10a11,16
>  * Modification History
>  *
>  * Date     Who     Comment
>  * ----------------------------------------------------
>  * 17Aug00  EJB     Added Pseudo-socket communications commands
>  * 23Aug00  DJS     Added include of os_xts
18a25,31
> #ifdef _XTS
> # include "os_xts.h"
> # ifdef USE_P_SOCKET
>   extern int my_fd;                /* Pseudo-socket file descriptor */
> # endif
> #endif
>
200c213
<                             fputs(buf, stdout);
---
>                             PSOUT(buf, stdout);
878c891
<     (void) fflush(stdout);
---
>     (void) PFLUSH(stdout);

File:  sendmail-8.9.3/src/sendmail.h
12a13,17
>  *
>  * Modification History
>  * Date     Who     Comment
>  * ----------------------------------------------------
>  * 23Aug00  DJS     Don't include os_xts.h, include it when needed in
.c files
45a51
> # ifndef _XTS
46a53
> # endif /*_XTS*/
```

```
File:  sendmail-8.9.3orig/src/srvrsmtp.c
/usr2/shifflet/wip/sendmail/sendmail-8.9.3/src/srvrsmtp.c
10a11,17
>  * Modification History
>  *
>  * Date      Who      Comment
>  * ----------------------------------------------------
>  * 17Aug00   EJB      Added Pseudo-socket communications commands
>  *           DJS      and debugging script
>  * 23Aug00   DJS      Added include of os_xts
22a30,36
> #ifdef _XTS
> # include "os_xts.h"
> # ifdef USE_P_SOCKET
>   extern int my_fd;                /* Pseudo-socket file descriptor */
> # endif
> #endif
>
221c235
<           (void) fflush(stdout);
---
>           (void) PFLUSH(stdout);
473a488,500
> #ifdef _XTS
> FILE *ctmpf = NULL;               /* DJS - debugging file descriptor
*/
> char ctmpf_out[1000];
>    sprintf (ctmpf_out,"/tmp/smtp_%d.tmp",getpid());
>    ctmpf = fopen (ctmpf_out,"a+");
>    FILE *tmpf2 = freopen (ctmpf_out,"a",stdout);
>    TMP_OUT_0("hello world\n", ctmpf);
>    TMP_OUT_1(ctmpf_out, "sendmail started [%d]\n", getpid(), ctmpf);
> #ifdef USE_P_SOCKET
> // change PID in PSKT so the child gets the signal
> pskt_set_aps_pid(my_fd, getpid());
> #endif
> #endif
494a522,524
> #ifdef _XTS
>    TMP_OUT_0("calling initsys\n", ctmpf);
> #endif
495a526,528
> #ifdef _XTS
>    TMP_OUT_0("initsys is done\n", ctmpf);
> #endif
1424a1458
> printf("Waiting for child exit [%d]\n", childpid);
1425a1460,1466
> printf("Wait returns [%d]\n", st);
> #ifdef _XTS
> #ifdef USE_P_SOCKET
> // change PID in PSKT so the parent gets the signal
> pskt_set_aps_pid(my_fd, getpid());
> #endif
> #endif

File:  sendmail-8.9.3/src/util.c
```

```
10a11,16
>  * Modification History
>  *
>  * Date     Who     Comment
>  * ----------------------------------------------------------
>  * 17Aug00  EJB     Added Pseudo-socket communications commands
>  * 23Aug00  DJS     Added include of os_xts
18a25,31
>
> #ifdef _XTS
> # include "os_xts.h"
> # ifdef USE_P_SOCKET
>   extern int my_fd;                /* Pseudo-socket file descriptor */
> # endif
> #endif
490c503
<                 (void) putchar(' ');
---
>                 (void) PBOUT(' ');
493c506
<     (void) putchar('\n');
---
>     (void) PBOUT('\n');
563c576
<                             putchar('&');
---
>                             PBOUT('&');
568c581
<                             putchar(*s++);
---
>                             PBOUT(*s++);
604c617
<                 putchar(c);
---
>                 PBOUT(c);
630,631c643,644
<                 (void) putchar('\\');
<                 (void) putchar(c);
---
>                 (void) PBOUT('\\');
>                 (void) PBOUT(c);
635,636c648,649
<                 (void) putchar('^');
<                 (void) putchar(c ^ 0100);
---
>                 (void) PBOUT('^');
>                 (void) PBOUT(c ^ 0100);
641c654
<     (void) fflush(stdout);
---
>     (void) PFLUSH(stdout);
1063c1076
<     while (!feof(fp) && !ferror(fp))
---
>     while (!PFEOF(fp) && !PFERROR(fp))
1066c1079
<         p = fgets(buf, siz, fp);
```

```
---
>           p = PSIN_RET(buf, siz, fp);
1069c1082
<           clearerr(fp);
---
>           PCLEARERR(fp);


File:  sendmail-8.9.3/BuildTools/bin/Build
21a22,30
> #
> # MODIFICATION History:
> # Date     Who                Comments
> # 01Mar00  David Shifflett    Added XTS-300 specific changes
> #                             Added special link command for XTS-300
> # 08Aug00  David Shifflett    Added creation of the Pseudo-socket
version
> #                             of the makefile
> #
>
23a33,35
> # default link command
> LN="ln -s"
>
205a218,227
>
> if [ "$os" = "$node" -a "$arch" = "i486" -a "`uname -v`" = "STOP" ]
> then
>     # XTS-300 sets uname -s the same as uname -n
>     os=XTS
>     rel=`echo $rel | sed -e 's/\.$//'`
>     # XTS-300 link command, no symbolic links :(
>     LN="ln"
> fi
>
432c454
<     (cd $obj; ln -s ../*.[ch158] .)
---
>     (cd $obj; $LN ../*.[ch158] .)
435c457
<           (cd $obj; ln -s ../sendmail.hf .)
---
>           (cd $obj; $LN ../sendmail.hf .)
500a523,525
>
> # Now Make Pseudo-socket version of Makefile
>     sed -e 's/_XTS/_XTS -DUSE_P_SOCKET/' $obj/Makefile >
$obj/Makefile.pskt
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B: SENDMAIL CONFIGURATION FILE FOR THE XTS 300

```
######################################################################
######################################################################
###                  SENDMAIL CONFIGURATION FILE
###                          06/26/2000
###
######################################################################
######################################################################
# parent domain
DPnps.navy.mil

#my domain
DNastro.cs.$P

# my short hostname
Dwholmes

# my full hostname
Dj$w.$m
#Dj$w
#DJ$w.$D
DRrelayhost.holmes.$N
DVsimple
Dnmailer-daemon
DlFrom $g $d remote from $U
Do@.%
Dq$?x$x <$g>$|$g$.
De$j Sendmail $v/$V ready at $b
Odbackground
Om
OF0644
Og1
O AliasFile=/usr2/shifflet/wip/sendmail/sendmail-8.9.3/src/aliases
Oh/usr2/shifflet/wip/sendmail/sendmail-8.9.3/src/sendmail.hf
OL6
Oo
OQ/usr/mail
Or1h
OS/usr2/shifflet/wip/sendmail/sendmail-8.9.3/src/sendmail.st
OT3d
OU1
Ox8
Ox12
Pfirst-class=0
Pspecial-delivery=100
Pjunk=-100
Tnetwork
H?F?From: $q
H?D?Date: $a
H?M?Message-Id: <p.$t@$j>
HSubject:
```

```
S0
R$*@$j      $#local$:$1 optional
R$*@$w      $#local$:$1 optional
R$-   $#local$:$1 optional
R$*   $#remote$@$R$:$1
S1
S2
S3
R$*<$+>$*   $2
S4
Mremote, P=[IPC], F=nsmFDMuXC, \
        S=10, R=10,A=IPC $h
Mlocal, P=/bin/mail, F=lsDFmn, S=10, R=10, A=mail -s $u
# Mlocal, P=/bin/mail, F=lsDFrmn, S=10, R=10, A=mail -r $f -d $u
Mprog, P=/bin/echo, F=lsDFMmn, S=10, R=10, A=mail $u
```

LIST OF REFERENCES

1. Costales, Bryan and Allman, Eric, *Sendmail*, O'Reilly and Associates, *Sebastopol, CA, 1997.*

2. Wilson, J. D., "Multilevel Secure Local Area Network Project Protocol Requirements Document," Masters Thesis, Naval Postgraduate School, Monterey, CA, June 2000.

3. *Guidance for Applying the Department of Defense Trusted Computer System Evaluation Criteria in Specific Environments*, CSC-STD-004-85, 25 June 1985.

4. *Department of Defense Trusted Computer System Evaluation Criteria*, DoD 5200.28-STD, December 1985.

5. *Common Criteria for Information Technology Security Evaluation Version 2.1*, Common Criteria Project Sponsoring Organizations, August 1999.

6. Bryer-Joyner, S. and Heller, Scott D., "*Secure Local Area Network Services for a High Assurance Multilevel Network*," Masters Thesis, Naval Postgraduate School Monterey, CA, March 1999.

7. Eads, Bradley R., "*Developing a High Assurance Multilevel Mail Server*," Masters Thesis, Naval Postgraduate School, Monterey, CA, March 1999.

8. The IMAP Connection – What is IMAP? http://www.imap.Org/about/whatisIMAP.html.

9. Wang Government Services, XTS-300 Users Manual, McLean, VA, March 1998, Stop Version 4.4.2.

10. Bell, D. E. and LaPadula, L. J. *Secure Computer System: Unified Exposition and Multics Interpretation, ESD-TR-75-306, Electronic Systems Division (AFSC) 1976.*

11. Biba, K. J. Integrity Considerations for Secure Computer Systems, MTR-3153, Mitre Corporation, Bedford, MA, April 1977.

12. Avolio, Frederick M. and Vixie, Paul A. *Sendmail, Theory and Practice, Digital Press, Boston, MA, 1997.*

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

No. copies

1. Defense Technical Information Center 2
   8725 John J. Kingman Road, Suite 0944
   Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library 2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, CA 93943-5101

3. Chairman, Code CS 1
   Naval Postgraduate School
   Monterey, CA 93943-5101

4. Dr. Cynthia E. Irvine 3
   Computer Science Department, Code CS/Ic
   Naval Postgraduate School
   Monterey, CA 93943-5000

5. Mr. James P. Anderson 1
   James P. Anderson Company
   Box 42
   Fort Washington, PA 19034

6. Mr. Paul Pitelli 1
   National Security Agency
   Research and Development Building
   R2, Technical Director
   9800 Savage Road
   Fort Mead, MD 20755-2600

7. LT Emma Brown 2
   5723 Goldenrod Road
   Albany, GA 31707

8. Mr. Richard Hale 1
   Defense Information Systems Agency
   5600 Columbia Pike, Suite 400
   Falls Church, VA 22041-3230

9. Ms. Barbara Flemming 1
   Defense Information Systems Agency
   5600 Columbia Pike, Suite 400
   Falls Church, VA 22041-3230

10. Carl Siel 1
    Space and Naval Warfare Systems Command
    PMW 161
    Building OT-1, Room 1024
    4301 Pacific Highway
    San Diego, CA 92110-3127

11. Commander, Naval Security Group Command 1
Naval Security Group Headquarters
9800 Savage Road
Suite 6585
Fort Meade, MD 20755-6585

12. Ms. Deborah M. Cooper 1
Deborah M. Cooper Company
P.O. Box 17753
Arlington, VA 22216

13. Ms. Louise Davidson 1
N643
Presidential Tower 1
2511 South Jefferson Davis Highway
Arlington, VA 22202

14. Mr. William Dawson 1
Community CIO Office
Washington, DC 20505

15. Capt James Newman.....1
N64
Presidential Tower 1
2511 South Jefferson Davis Highway
Arlington, VA 22202

16. Mr. James Knoke 1

Wang Government Services, Inc.

7900 Westport Dr.

McClean, VA 22102-4299

17. Mr. Michael Focke 1

Wang Government Services, Inc.

7900 Westport Dr.

McClean, VA 22102-4299